Intel® Xeon™ Processor

Specification Update

October 2002

Notice: The Intel® Xeon™ processor may contain design defects or errors known as errata which may cause the product to deviate from published specifications. Current characterized errata are documented in this Specification Update.

Document Number:249678-019

Contents

Revision History

Version	Description	Date
-001	Initial Release.	May 2001
-002	Added errata P27-P28.	June 2001
-003	Updated erratum P25.	June 2001
-004	Added RCPPS, RCPSS, RSQRTPS and RSQRTSS instruction specification clarification.	July 2001
-005	Added errata P29-P32. Added Unused outputs specification clarifications.	August 2001
-006	Added errata P33-34. Production mark update to include 2D matrix	September 2001
-007	Added 2 GHz frequency, D0 stepping and FC-BGA package information Added DP Platform Population Matrix Updated Summary of Errata table with applicable “Fixed” & “No Fix” errata plans Updated Errata P32	October 2001
-008	Added Erratum P35 Updated Summary of Errata Table with latest errata status Added Documentation Changes P1-P5	November 2001
-009	Added errata P36 and P37. Added Documentation Change P6. Updated Summary Tables with latest errata and documentation changes.	December 2001
-010	Added Documentation Changes P7-P11. Added Errata P38-P44. Added new processor with CPUID=0F24h B0 Step. Added new Sspec processors to Processor ID and DP Matrix tables. Added new package markings(Figs 3,4) for 512KB Cache processor.	January 2002
-011	Corrected 3 mislabled S-Spec Table parts.	January 2002
-012	Added Erratum P45 to Processor ID Table.	February 2002
-013	Added Errata P46 and P47. Added Documentation Changes P1-P6. Minor changes to “Fix” descriptions.	April 2002
-014	Added PWRGOOD Specification Change. Added Errata P48. Updated Errata P11. Added Specification Changes P1-P3. Added Documentation Changes P1-P3. Added new S-specs SL687 and SL65T to Processor ID table. Updated datasheet name to 2.40GHz	May 2002
-015	Added Erratum P49. Updated Errata P24, P40 status Added Document Changes P1-P2	June 2002

Version	Description	Date
-016	Added new Erratum P50. Added new Doc Changes P3-P12. Edited Summary of Errata Table erratum P40 to PlanFix. Minor edits to processor markings.	July 2002
-017	Edited DP Matrix table. Updated the Summary of Errata table w C1 Step info.	August 2002
-018	Added errata P51, P52. Edited erratum P14. Added Doc Changes P3-P24. Added Spec Clarification P1. Added new C1 S-specs to Processor ID Info table.	September 2002
-019	Edited erratum P35. Removed erratum previously numbered P52 as not applicable. Added new erratum P52. Added Doc Changes P25-P32.	October 2002

Preface

This document is an update to the specifications contained in the documents listed in the following Affected Documents/Related Documents table. It is a compilation of device and document errata and specification clarifications and changes, and is intended for hardware system manufacturers and for software developers of applications, operating system, and tools.

Information types defined in the Nomenclature section of this document are consolidated into this update document and are no longer published in other documents. This document may also contain information that has not been previously published.

Affected Documents/Related Documents

Document Title	Document Number
Intel® Xeon™ Processor at 1.40 GHz, 1.50 GHz, 1.70 and 2 GHz datasheet	249665
Intel® Xeon™ Processor with 512 KB L2 Cache at 1.80 GHz to 2.80 GHz Datasheet	298642
IA-32 Intel® Architecture Software Developer's Manual, Volumes 1, 2 and 3	245470, 245471, and 245472, respectively

Nomenclature

S-Spec Number is a five-digit code used to identify products. Products are differentiated by their unique characteristics, e.g., core speed, L2 cache size, package type, etc., as described in the processor identification information table. Care should be taken to read all notes associated with each S-Spec number.

Errata are design defects or errors. Errata may cause the Intel Xeon processor's behavior to deviate from published specifications. Hardware and software designed to be used with any given stepping must assume that all errata documented for that stepping are present on all devices.

Specification Changes are modifications to the current published specifications. These changes will be incorporated in the next release of the specifications.

Specification Clarifications describe a specification in greater detail or further highlight a specification's impact to a complex design situation. These clarifications will be incorporated in the next release of the specifications.

Documentation Changes include typos, errors, or omissions from the current published specifications. These changes will be incorporated in the next release of the specifications.

Intel® Xeon™ Processor Markings, 256-KB Cache (603-pin interposer INT-mPGA package)

Figure 1. Top Side Processor Marking

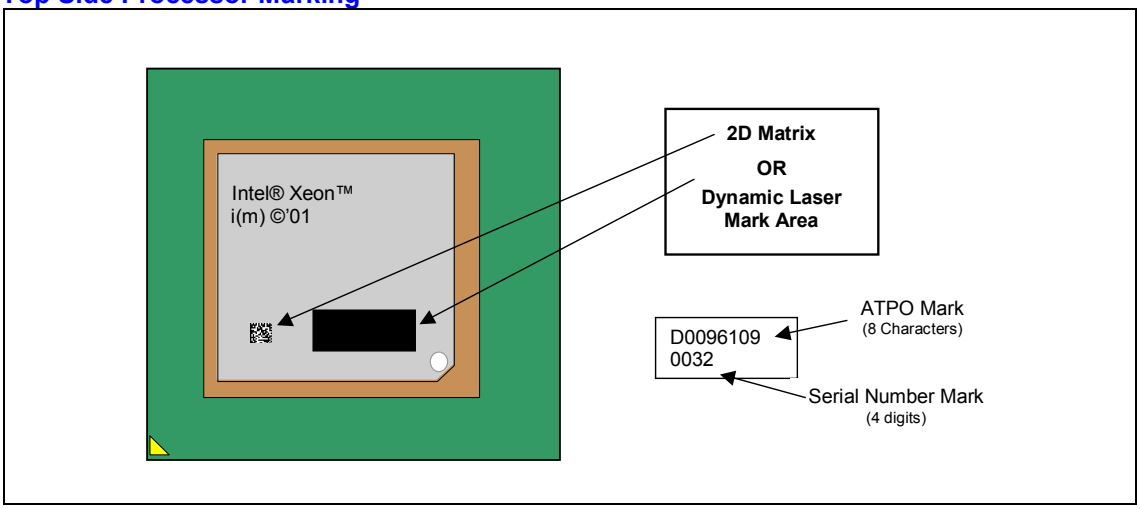
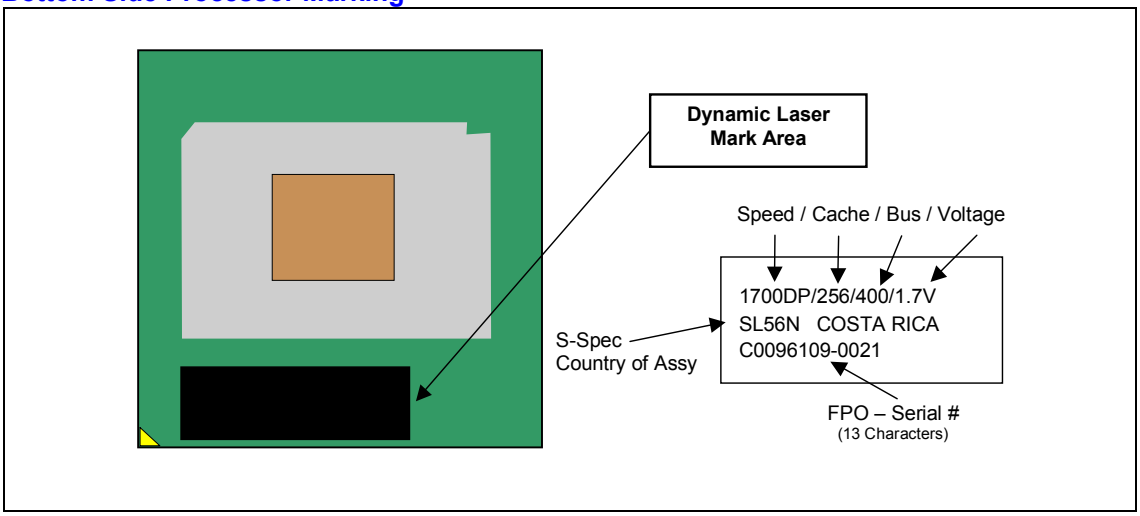


Figure 2. Bottom Side Processor Marking



Intel® Xeon™ Processor Markings, 512-KB Cache (603-pin interposer INT-mPGA package)

Figure 3. Top Side Processor Marking

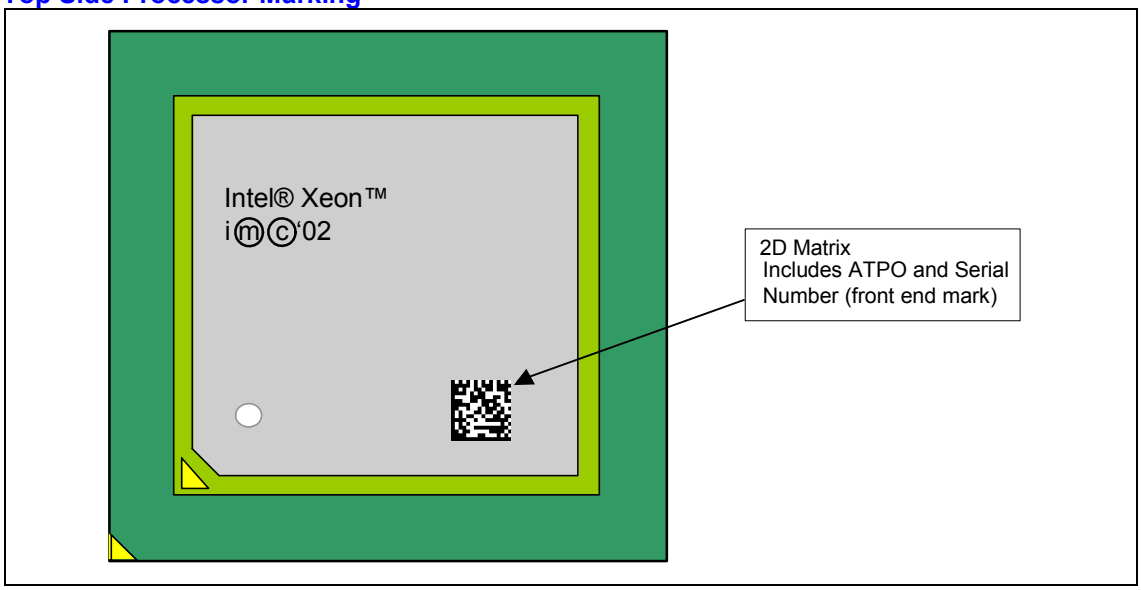


Figure 4. Bottom Side Processor Marking

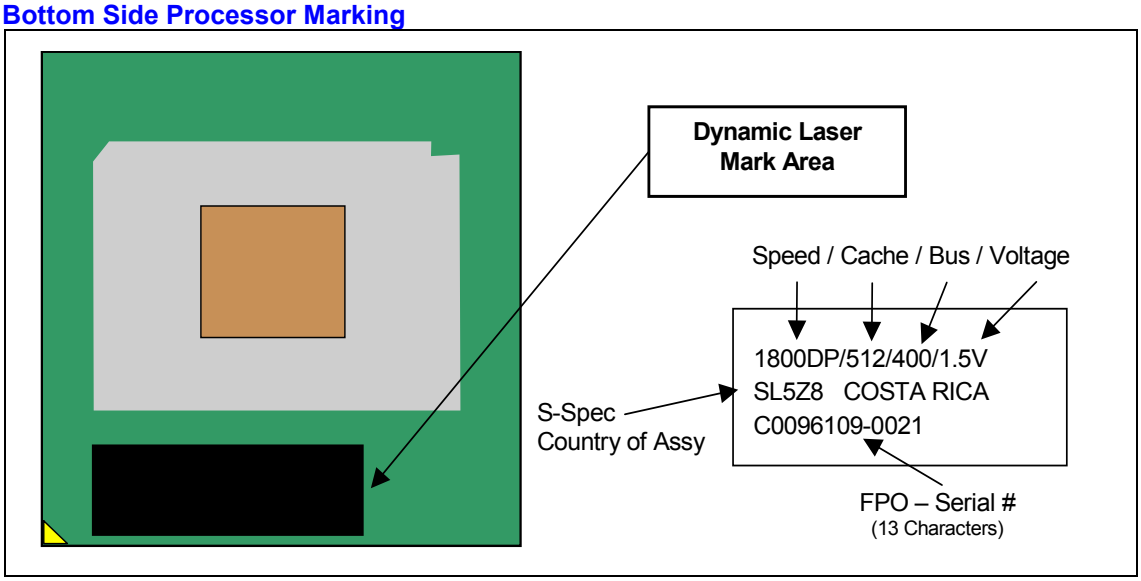


Figure 5. Example of Production Mark – Top View

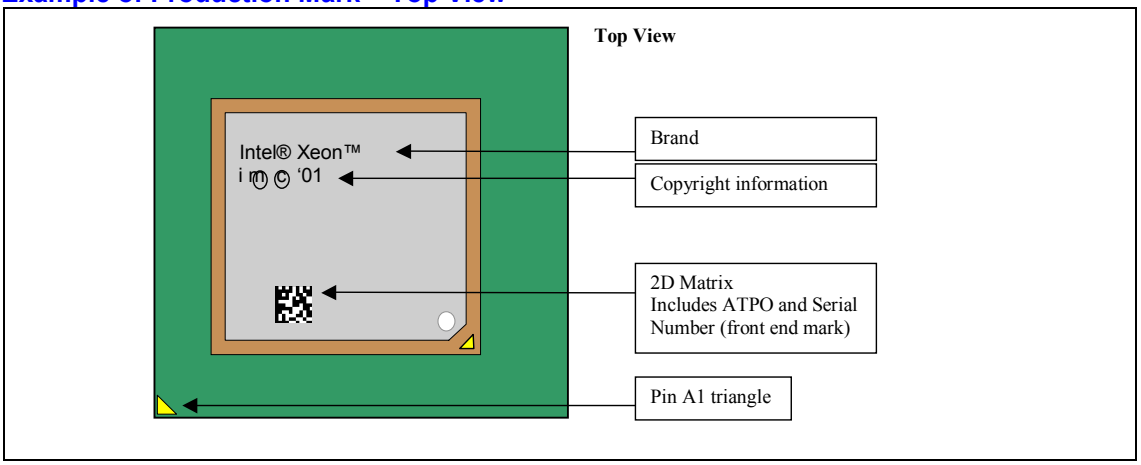
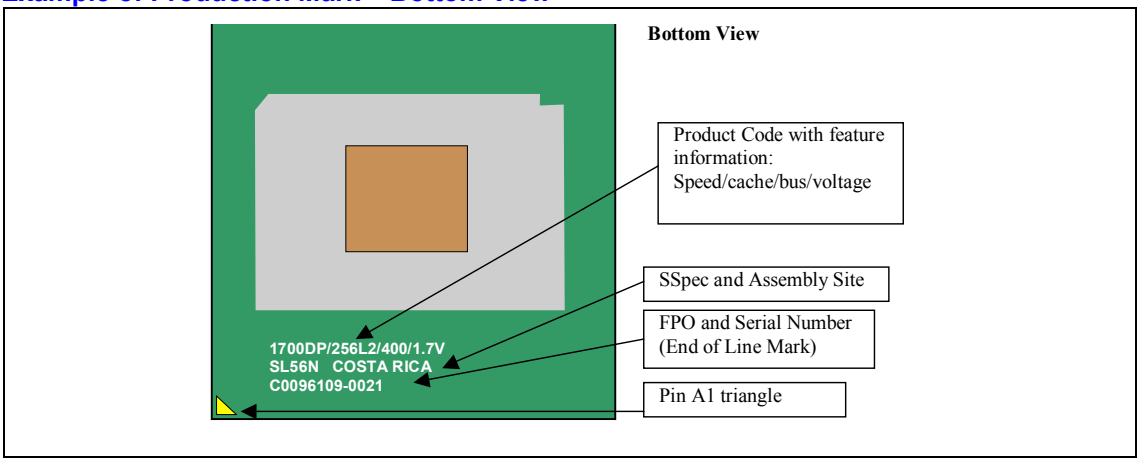


Figure 6. Example of Production Mark – Bottom View



Identification Information

The Intel Xeon processor can be identified by the following values:

Family[1]	Model[2]	Brand ID[3]
1111	0000	00001110
1111	0001	00001110
1111	0010	00001011

NOTES:

1. The Family corresponds to bits [11:8] of the EDX register after RESET, bits [11:8] of the EAX register after the CPUID instruction is executed with a 1 in the EAX register, and the generation field of the Device ID register accessible through Boundary Scan.
2. The Model corresponds to bits [7:4] of the EDX register after RESET, bits [7:4] of the EAX register after the CPUID 2 instruction is executed with a 1 in the EAX register, and the model field of the Device ID register accessible through Boundary Scan.
3. The Brand ID corresponds to bits [7:0] of the EBX register after the CPUID instruction is executed with a 1 in the EAX register.

Cache and TLB descriptor parameters are provided in the EAX, EBX, ECX and EDX registers after the CPUID instruction is executed with a 2 in the EAX register. Please refer to the Intel Processor Identification and the CPUID Instruction Application Note (AP-485) for further information on the CPUID instruction.

Table 1. Intel® Xeon™ Processor Identification and Package Information

S-Spec Number	Core Stepping	CPUID	Speed Core/System Bus (GHz/MHz)	L2 Size (Kbytes)	Processor Interposer Revision	Package and Revision	Notes
SL4WX	C1	0F0Ah	1.40/400	256K	B0	31 mm OLGA rev 2.0	1, 4
SL56G	C1	0F0Ah	1.40/400	256K	B0	31 mm OLGA rev 2.0	1, 2, 4
SL4WY	C1	0F0Ah	1.50/400	256K	B0	31 mm OLGA rev 2.0	1, 4
SL4ZT	C1	0F0Ah	1.50/400	256K	B0	31 mm OLGA rev 2.0	1, 2, 4
SL56N	C1	0F0Ah	1.70/400	256K	B0	31 mm OLGA rev 2.0	1, 4
SL56H	C1	0F0Ah	1.70/400	256K	B0	31 mm OLGA rev 2.0	1, 2, 4
SL5TD	D0	0F12h	1.50/400	256K	C0	31 mm FC-BGA rev 3.0	1, 4
SL5U6	D0	0F12h	1.50/400	256K	C0	31 mm FC-BGA rev 3.0	1, 2, 3, 4
SL5TE	D0	0F12h	1.70/400	256K	C0	31 mm BC-BGA rev 3.0	1,4
SL5U7	D0	0F12h	1.70/400	256K	C0	31 mm BC-BGA rev 3.0	1, 2, 3, 4
SL5TH	D0	0F12h	2/400	256K	C0	31 mm FC-BGA rev 3.0	1, 3, 4
SL5U8	D0	0F12h	2/400	256K	C0	31 mm FC-BGA rev 3.0	1, 2,3,4
SL5Z8	B0	0F24h	1.80/400	512K	01	35 mm FC-BGA	1
SL622	B0	0F24h	1.80/400	512K	01	35 mm FC-BGA	1, 2
SL5Z9	B0	0F24h	2/400	512K	01	35 mm FC-BGA	1
SL623	B0	0F24h	2/400	512K	01	35 mm FC-BGA	1, 2
SL5ZA	B0	0F24h	2.20/400	512K	01	35mm FC-BGA	1
SL624	B0	0F24h	2.20/400	512K	01	35mm FC-BGA	1, 2
SL687	B0	0F24h	2.40/400	512K	01	35 mm FC-BGA	1,2
SL65T	B0	0F24h	2.40/400	512K	01	35 mm FC-BGA	
SL6EL	C1	0F27H	1.80/400	512k	01	35 mm FC-BGA	
SL6EM	C1	0F27H	2/400	512K	01	35 mm FC-BGA	
SL6EN	C1	0F27H	2.2/400	512K	01	35 mm FC-BGA	
SL6EP	C1	0F27H	2.4/400	512K	01	35 mm FC-BGA	
SL6EQ	C1	0F27H	2.6/400	512K	01	35 mm FC-BGA	
SL6K3	C1	0F27H	2.6/400	512K	01	35 mm FC-BGA	2
SL6M7	C1	0F27H	2.8/400	512K	01	35 mm FC-BGA	
SL6MS	C1	0F27H	2.8/400	512K	01	35 mm FC-BGA	2

NOTES:

1. The Intel Xeon processor listed here is installed onto a micro pin grid array (mPGA) interposer. The overall processor package is called INT-mPGA.
2. These parts are Intel boxed processors.
3. FC-BGA packaging maintains form, fit, and functionality when compared to OLGA packaging. Users may notice a color change.
4. These parts require the inputs from A20M#, IGNNE#, LINT[1]/NMI and LINT[0]/INTR pins during RESET to set the correct core to bus frequency ratio.

Mixed Steppings in DP Systems

Intel Corporation fully supports mixed steppings of Intel® Xeon™ processors. The following list and processor matrix describes the requirements to support mixed steppings:

- Mixed steppings are only supported with processors that have identical family numbers as indicated by the CPUID instruction. The Intel® Xeon™ processor is available with two different Model numbers as indicated by the CPUID. Please refer to the "DP Platform Population Matrix for the Intel® Xeon™ Processor" for details regarding inclusion of processors with mixed CPUID/Core steppings.
- While Intel has done nothing to specifically prevent processors operating at differing frequencies from functioning within a multiprocessor system, there may be uncharacterized errata that exist in such configurations. Intel does not support such configurations. In mixed stepping systems, all processors must operate at identical frequencies (i.e., the highest frequency rating commonly supported by all processors).
- While there are no known issues associated with the mixing of processors with differing cache sizes in a multiprocessor system, and Intel has done nothing to specifically prevent such system configurations from operating, Intel does not support such configurations since there may be uncharacterized errata that exist. In mixed stepping systems, all processors must be of the same cache size.
- While Intel believes that certain customers may wish to perform validation of system configurations with mixed frequency or cache sizes, and that those efforts are an acceptable option to our customers, customers would be fully responsible for the validation of such configurations.
- Intel requires that the proper microcode update be loaded on each processor operating in a multiprocessor system. Any processor that does not have the proper microcode update loaded is considered by Intel to be operating out of specification.
- The workarounds identified in this and following specification updates must be properly applied to each processor in the system. Certain errata are specific to the multiprocessor environment and are identified in the *Mixed Stepping Processor Matrix* found at the end of this section. Errata for all processor steppings will affect system performance if not properly worked around. Also see the Intel® Xeon™ Processor Identification and Package Information section for additional details on which processors are affected by specific errata.
- In mixed stepping systems, the processor with the lowest feature-set, as determined by the CPUID Feature Bytes, must be the Bootstrap Processor (BSP). In the event of a tie in feature-set, the tie should be resolved by selecting the BSP as the processor with the lowest stepping as determined by the CPUID instruction.

In the following processor matrix, **"NI"** indicates that there are currently no known issues associated with mixing these steppings. A number indicates that a known issue has been identified as listed in the table following the matrix. **"X"** indicates the processors cannot be mixed. A dual processor system using mixed processor steppings must assure that errata are addressed appropriately for each processor.

Table 2. DP Platform Population Matrix for the Intel® Xeon™ Processor

Processor Signature/Core Stepping	0F0Ah/C1	0F12h/D0	0F24h/B0	0F27h/C1
0F0Ah/C1	NI	Note 1	X	X
0F12h/D0	Note 1	NI	X	X
0F24h/B0	X	X	NI	NI
0F27h/C1	X	X	NI	NI

NOTES:

1. Some of these processors are affected by errata, which may affect the features an MP system is able to support. See the *Intel® Xeon™ Processor Identification and Package Information* table for details on which processors are affected by these errata.

Summary Tables of Changes

The following table indicates the Specification Changes, Errata, Specification Clarifications or Documentation Changes, which apply to the listed MCH steppings. Intel intends to fix some of the errata in a future stepping of the component, and to account for the other outstanding issues through documentation or Specification Changes as noted. This table uses the following notations:

Codes Used in Summary Table

X:	Erratum, Specification Change or Clarification that applies to the given processor stepping.
(No mark) or (Blank Box):	This erratum is fixed in listed stepping or specification change does not apply to listed stepping.
Doc:	Document change or update that will be implemented.
PlanFix:	This erratum may be fixed in a future of the product.
Fixed:	This erratum has been previously fixed.
NoFix:	There are no plans to fix this erratum.
PKG:	This column refers to errata on the Intel Xeon processor substrate
AP	APIC-related erratum
Shaded:	This item is either new or modified from the previous version of the document.

Each Specification Update item is prefixed with a capital letter to distinguish the product. The key below details the letters that are used in Intel's microprocessor Specification Updates:

A = Intel® Pentium® II processor
B = Mobile Intel® Pentium® II processor
C = Intel® Celeron® processor
D = Intel® Pentium® II Xeon™ processor
E = Intel® Pentium® III processor
G = Intel® Pentium® III Xeon™ processor
H = Mobile Intel® Celeron® processor at 466 MHz 433 MHz, 400 MHz, 366 MHz, 333 MHz, 300 MHz, and 266 MHz
K = Mobile Intel® Pentium® III Processor - M
M = Mobile Intel® Celeron® processor at 500 MHz, 450 MHz, and 400A MHz
N = Intel® Pentium® 4 processor
P = Intel® Xeon™ processor and Intel® Xeon™ processor with 512-KB L2 Cache
T = Mobile Intel® Pentium® 4 processor
V = Mobile Intel® Celeron® processor on 0.13 Micron Process in Micro-FCPGA Package
W = Low Voltage Intel® Xeon™ processor"

Note: The Specification Updates for the Pentium® processor, Pentium® Pro processor, and other Intel products do not use this convention.

No.	Steppings				Plans	ERRATA
	C1/ 0F0Ah	D0/ 0F12h	B0/ 0F24h	C1/ 0F27h		
P1	X	X			Fixed	UC Code in same line as WriteBack (WB) data may lead to data corruption
P2	X	X	X	X	No Fix	Transaction is not retried after BINIT#
P3	X	X	X	X	No Fix	Invalid opcode 0FFFH requires a ModRM byte
P4	X	X	X	X	No Fix	When in No-Fill Mode (CR0.CD=1) the memory type of large (PSE-4M and PAE-2M) pages are wrongly forced to uncacheable
P5	X	X	X	X	No Fix	Processor may hang due to Speculative Page Walks to Non-Existent System Memory
P6	X	X			Fixed	Writing a performance counter may result in an incorrect counter value
P7	X	X			Fixed	Performance Counter May Contain Incorrect Value After Being Stopped
P8	X				Fixed	REP MOV instruction with overlapping source and destination may result in data corruption
P9	X	X	X	X	No Fix	Memory type of the load lock different from its corresponding store unlock
P10	X	X	X	X	No Fix	Machine check architecture error reporting and recovery may not work as expected
P11	X	X	X	X	No Fix	Debug mechanisms may not function as expected
P12	X				Fixed	Processor may live-lock if PDEs or PTEs are in UC space
P13	X				Fixed	Thermal status log bit may not be set when the Thermal Control Circuit is Active
P14	X	X			Fixed	Processor may timeout waiting for a device to respond after 0.67 seconds
P15	X	X	X	X	No Fix	Cascading of performance counters does not work correctly when forced overflow is enabled
P16	X	X	X	X	No Fix	EMON event counting of X87 loads may not work as expected
P17	X	X			Fixed	Simultaneous code breakpoint and uncorrectable error results in a processor hang
P18	X	X			Fixed	Software controlled clock modulation using a 12.5% or 25% duty cycle may cause the processor to hang
P19	X				Fixed	RFO with ECC error may result in data corruption
P20	X				Fixed	Speculative page fault may cause livelock
P21	X				Fixed	PAT index MSB may be calculated incorrectly
P22	X	X	X	X	No Fix	System bus interrupt messages without data and which receive a HardFailure response may hang the processor
P23	X	X			Fixed	SQRTPD and SQRTSD May Return QnaN Indefinite Instead of Negative Zero
P24	X	X			Fixed	Bus Invalidate Line Request that return unexpected data may result in L1 cache corruption
P25	X				Fixed	Multi-processor boot protocol may not complete with an IOQ depth of one

No.	Steppings				Plans	ERRATA
	C1/ 0F0Ah	D0/ 0F12h	B0/ 0F24h	C1/ 0F27h		
P26	X	X	X	X	No Fix	Processor flags #PF instead of #AC on an unlocked CMPXCHG8B instruction
P27	X	X	X		Fixed	Incorrect data may be returned when page tables are located in Write Combining (WC) memory
P28	X	X	X	X	No Fix	FSW may not be completely restored after page fault on FRSTOR or FLDENV instructions
P29	X	X	X		Fixed	Write Combining (WC) load may result in bogus address on system bus
P30	X	X	X	X	No Fix	Processor provides a 4-byte store unlock after an 8-byte load lock
P31	X	X	X		Fixed	Multiple accesses to the same S-state L2 cache line and ECC error combination may result in loss of cache coherency
P32	X	X	X	X	No Fix	IA32_MC0_ADDR and IA32_MC0_MISC registers will contain invalid or stale data following a data, address, or response parity error
P33	X	X	X	X	No Fix	When the processor is in the System Management Mode (SMM), Debug Registers may be fully writeable
P34	X	X	X	X	No Fix	Associated counting logic must be configured when using Event Selection Control (ESCR) MSR
P35	X	X	X		Fixed	Livelock may occur when bus parking is disabled
P36	X	X	X		Fixed	CR2 may be incorrect or an incorrect page fault error code may be pushed onto stack after execution of an LSS instruction
P37	X	X			Fixed	Buffer On Resistance may exceed specification
P38			X		Fixed	Instruction pointer stored on stack may become invalid
P39			X	X	No Fix	Shutdown and IERR# may result due to a Machine Check Exception on a Hyper-Threading technology enabled processor
P40			X		Fixed	Hyper-Threading technology enabled processors may hang in the presence of extensive self-modifying code
P41			X		Fixed	Global bit incorrectly set for secondary logical processors in ITLB
P42			X		Fixed	Machine Check Exception (MCE) observed on DP Platforms
P43			X	X	PlanFix	BPM[5:3]# V_{IL} does not meet specification
P44			X	X	No Fix	Processor may hang under certain frequencies and 12.5% STPCLK# duty cycle
P45	X	X	X	X	No Fix	System may hang if a fatal cache error causes Bus Write Line (BWL) transaction to occur to the same cache line address as an outstanding Bus Read Line (BRL) or Bus Read-Invalidate Line (BRIL)
P46	X	X	X		Fixed	L2 Cache may contain Stale Data in the Exclusive State
P47	X	X	X	X	PlanFix	Re-mapping the APIC Base Address to a value less than or equal to 0xDC001000 may cause IO and Special Cycle Failure
P48			X	X	PlanFix	Erroneous BIST result found in EAX Register after Reset

No.	Steppings				Plans	ERRATA
	C1/ 0F0Ah	D0/ 0F12h	B0/ 0F24h	C1/ 0F27h		
P49	X	X	X		Fixed	Processor does not Flag #GP on no-zero write to certain MSRs
P50	X	X	X	X	No Fix	Simultaneous assertion of A20M# and INIT# may result in incorrect data fetch
P51			X	X	Plan Fix	Processor does not respond to Break Requests from ITP
P52	X	X	X		Fixed	CPUID instruction returns incorrect number of ITLB entries

Number	Plan	SPECIFICATION CHANGES
P1	PlanFix	BR[3:2]# Signal Definition is not Reserved

Number	SPECIFICATION CLARIFICATIONS
	There are no Specification Clarifications in this Specification Update revision.

Number	Plans	DOCUMENTATION CHANGES
P1	Doc	SSE and SSE2 instructions opcodes
P2	Doc	Executing the SSE2 variant on a non-SSE2 capable processor
P3	Doc	Direction Flag (DF) Mistakenly Denoted as a System Flag
P4	Doc	Fopcode Compatibility Mode
P5	Doc	FCOS, FPTAN, FSIN, and FSINCOS Trigonometric Domain not Correct
P6	Doc	Incorrect Description in top of stack
P7	Doc	EFLAGS Register Correction
P8	Doc	PSE-36 Paging Mechanism
P9	Doc	0x33 Opcode
P10	Doc	Incorrect Information for SLDT
P11	Doc	LGDT/LIDT Instruction Information Correction
P12	Doc	Errors In Instruction Set Reference
P13	Doc	RSM Instruction Set Summary
P14	Doc	Correct MOVAPS and MOVAPD Operand Section
P15	Doc	DAA—Decimal Adjust AL after Addition
P16	Doc	DAS—Decimal Adjust AL after Subtraction
P17	Doc	Omission of Dependency between BTM and LBR
P18	Doc	I/O Permissions Bitmap Base Addy > 0xDFFF Does not Cause #GP(0) Fault
P19	Doc	Wrong Field Width for MINSS and MAXSS
P20	Doc	Figure 15-12 PEBS Record Format
P21	Doc	I/O Permission Bit Map
P22	Doc	Cache Description
P23	Doc	Instruction Formats and Encoding

P24	Doc	Machine-Check Initialization
P25	Doc	Incorrect Description of MOVAPS #UD
P26	Doc	PSE-36 Paging Mechanism
P27	Doc	Segment Wraparound compatibility
P28	Doc	Performance Counting Clocks
P29	Doc	New Cascading Counters Chapter
P30	Doc	Updated Table B-1 in Appendix B,Vol.3
P31	Doc	Message Signaled Interrupts
P32	Doc	Selecting Memory Types For Pentium 4, Intel Xeon, and Pentium III Processors

Errata

P1. UC Code in Same Line as WriteBack (WB) Data may Lead to Data Corruption

Problem: This erratum occurs when both code (being accessed as UC or WC) and data (being accessed as WB) are placed in the same cache line. The UC fetch will cause the processor to self-snoop and generate an implicit writeback. The data supplied by this implicit writeback may be corrupted due to the way the processor is currently handling self-modifying code.

Implication: UC code located in the same cache line as WB data may lead to data corruption.

Workaround: UC or WC code should not be located in the same 64 byte cache line as any location that is being stored to with WB data.

Status: For the steppings affected, see the *Summary Tables of Changes*.

P2. Transaction Is Not Retried After BINIT#

Problem: If the first transaction of a locked sequence receives a HITM# and DEFER# during the snoop phase it should be retried and the locked sequence restarted. However, if BINIT# is also asserted during this transaction, the transaction will not be retried.

Implication: When this erratum occurs, locked transactions will not be retried.

Workaround: None identified

Status: For the steppings affected, see the *Summary Tables of Changes*.

P3. Invalid Opcode 0FFFh Requires A ModRM Byte

Problem: Some invalid opcodes require a ModRM byte and other following bytes, while others do not. The invalid opcode 0FFFh did not require a ModRM in previous generation microprocessors such as Pentium® II or Pentium III processors, but it is required in the Intel® Xeon™ processor

Implication: The use of an invalid opcode 0FFFh without the ModRM byte may result in a page or limit fault on the Intel Xeon processor.

Workaround: To avoid this erratum use ModRM byte with invalid 0FFFh opcode.

Status: For the steppings affected, see the *Summary Tables of Changes*.

P4. When in No-Fill Mode (CR0.CD=1) the Memory Type of Large (PSE-4M and PAE-2M) Pages are Wrongly Forced to Uncacheable

Problem: When the processor is operating in No-Fill Mode (CR0.CD=1), the page miss hardware incorrectly forces the memory type of large (PSE-4M and PAE-2M) pages to UC memory type regardless of the MTRR settings. By forcing the memory type of these pages to UC, load operations, which should hit valid data in the L1 cache, are forced to load the data from system memory. Some applications will lose the performance advantage associated with the caching permitted by other memory types.

Implication: This erratum may result in some performance degradation when using no-fill mode with large pages.

Workaround: None identified

Status: For the steppings affected, see the *Summary Tables of Changes*.

P5. Processor May Hang Due to Speculative Page Walks to NonExistent System Memory

Problem: A load operation issued speculatively by the processor that misses the Data Translation Lookaside Buffer (DTLB) results in a page walk. A branch instruction older than the load retires so that this load operation is now in the mispredicted branch path. Due to an internal boundary condition, in some instances the load is not canceled before the page walk is issued.

The Page Miss Handler (PMH) starts a speculative page-walk for the Load and issues a cacheable load of the Page Directory Entry (PDE). This PDE load returns data that points to a page table entry in uncacheable (UC) memory. The PMH issues the PTE Load to UC space, which is issued on the system bus. No response comes back for this load PTE operation since the address is pointing to system memory, which does not exist.

This load to non-existent system memory causes the processor to hang because other bus requests are queued up behind this UC PTE load, which never gets a response. If the load was accessing valid system memory, the speculative page-walk would successfully complete and the processor would continue to make forward progress.

Implication: Processor may hang due to speculative page walks to non-existent system memory

Workaround: Page directories and page tables in UC memory space must point to system memory that exists.

Status: For the steppings affected, see the *Summary Tables of Changes*.

P6. Writing a Performance Counter May Result in an Incorrect Counter Value

Problem: Accessing a performance counter also enables the counter input so that writing one half of the counter can cause the other half to increment. When a performance counter is written and the event counter for the event being monitored is non-zero, the performance counter will be incremented by the value on that event counter. Because the upper eight bits of the performance counter are not written at the same time as the lower 32 bits, the increment due to the non-zero event counter may cause a carry to the upper bits such that the performance counter contains a value higher than what was written. The worst-case error caused by this can be about 4 billion counts.

Implication: When this erratum occurs, the performance counter will contain a different value from that which was written.

Workaround: If the performance counter is set to select a null event and the CCCR for that counter has its compare bit set to zero, before the performance counter is written, this erratum will not occur.

Status: For the steppings affected, see the *Summary Tables of Changes*.

P7. Performance Counter May Contain Incorrect Value After Being Stopped

Problem: If a performance counter is stopped on the precise internal clock cycle where the intermediate carry from the lower 32 bits of the counter to the upper eight bits occurs, the intermediate carry is lost.

Implication: When this erratum occurs the performance counter may contain a value about 4 billion (232) less than it should.

Workaround: Since this erratum does not occur if the performance counters are read when running, a possible workaround is to read the counter before stopping it.

Status: For the steppings affected, see the *Summary Tables of Changes*.

P8. REP MOV Instruction with Overlapping Source and Destination may Result in Data Corruption

Problem: When fast strings are enabled and a REP MOV instruction is used to move a string and the source and destination strings overlap by 56 bytes or less, data corruption may occur.

Implication: When this erratum occurs, data corruption may occur.

Workaround: It is possible for the BIOS to contain a workaround for this erratum.

Status: For the steppings affected, see the *Summary Tables of Changes*.

P9. Memory Type of the Load Lock Different from its Corresponding Store Unlock

Problem: The Intel® Xeon™ processor employs a use-once protocol to ensure that a processor in a multiprocessor system may access data that is loaded into its cache on a Read-for-Ownership operation at least once before it is snooped out by another processor. This protocol is necessary to avoid a dual processor livelock scenario where no processor in the system can gain ownership of a line and modify it before that data is snooped out by another processor. In the case of this erratum, the use-once protocol incorrectly activates for split load lock instructions. A load lock operation accesses data that splits across a page boundary with both pages of WB memory type. The use-once protocol activates and the memory type for the split halves get forced to UC. Since use-once does not apply to stores, the store unlock instructions go out as WB memory type. The full sequence on the Bus is: locked partial read (UC), partial read (UC), partial write (WB), locked partial write (WB). The Use-once protocol should not be applied to Load locks.

Implication: When this erratum occurs, the memory type of the load lock will be different than the memory type of the store unlock operation. This behavior (Load Locks and Store Unlocks having different memory types) does not however introduce any functional failures such as system hangs or memory corruption.

Workaround: None identified

Status: For the steppings affected, see the *Summary Tables of Changes*.

P10. Machine Check Architecture Error Reporting and Recovery May Not Work as Expected

Problem: When the processor detects errors it should attempt to report and/or recover from the error. In the situations described below, the processor does not report and/or recover from the error(s) as intended.

- When a transaction is deferred during the snoop phase and subsequently receives a Hard Failure response, the transaction should be removed from the bus queue so that the processor may proceed. Instead, the transaction is not properly removed from the bus queue, the bus queue is blocked, and the processor will hang.
- When a hardware prefetch results in an uncorrectable tag error in the L2 cache, MC0_STATUS.UNCOR and MC0_STATUS.PCC are set but no Machine Check Exception (MCE) is signaled. No data loss or corruption occurs because the data being prefetched has not been used. If the data location with the uncorrectable tag error is subsequently accessed, an MCE will occur. However, upon this MCE, or any other subsequent MCE, the information for that error will not be logged because MC0_STATUS.UNCOR has already been set and the MCA status registers will not contain information about the error which caused the MCE assertion but instead will contain information about the prefetch error event.
- When the reporting of errors is disabled for Machine Check Architecture (MCA) Bank 2 by setting all MC2_CTL register bits to 0, uncorrectable errors should be logged in the IA32_MC2_STATUS register but no machine-check exception should be generated. Uncorrectable loads on bank 2, which would normally be logged in the IA32_MC2_STATUS register, are not logged.
- When one half of a 64 byte instruction fetch from the L2 cache has an uncorrectable error and the other 32 byte half of the same fetch from the L2 cache has a correctable error, the processor will attempt to correct the correctable error but cannot proceed due to the uncorrectable error. When this occurs the processor will hang.
- When an L1 cache parity error occurs, the cache controller logic should write the physical address of the data memory location that produced that error into the IA32_MC1_ADDR REGISTER

(MC1_ADDR). In some instances of a parity error on a load operation that hits the L1 cache, however, the cache controller logic may write the physical address from a subsequent load or store operation into the IA32_MC1_ADDR register.

- The local xAPIC has an Error Status Register, which records all errors it detects. Bit 6 of this register, the Receive Illegal Vector bit, is set when the local xAPIC detects an illegal vector in a message that it received. When an illegal vector error is received on the same internal clock that the error status register is being written due to a previous error, bit 6 does not get set and illegal vector errors are not flagged.
- When an error exists in the tag field of a cache line such that a request for ownership (RFO) issued by the processor hits multiple tag fields in the L2 cache (the correct tag and the tag with the error) and the accessed data also has a correctable error, the processor will correctly log the multiple tag match error but will hang when attempting to execute the machine check exception handler.
- If a memory access receives a machine check error on both 64 byte halves of a 128-byte L2 cache sector, the IA32_MC0_STATUS register records this event as multiple errors, i.e., the valid error bit and the overflow error bit are both set indicating that a machine check error occurred while the results of a previous error were in the error-reporting bank. The IA32_MC1_STATUS register should also record this event as multiple errors but instead records this event as only one correctable error.
- The overflow bit should be set to indicate when more than one error has occurred. The overflow bit being set indicates that more than one error has occurred. Because of this erratum, if any further errors occur, the MCA overflow bit will not be updated, thereby incorrectly indicating only one error has been received.
- If an I/O instruction (IN, INS, REP INS, OUT, OUTS, or REP OUTS) is being executed, and if the data for this instruction becomes corrupted, the processor will signal a Machine Check Exception (MCE). If the instruction is directed at a device that is powered down, the processor may also receive an assertion of SMI#. Since MCEs have higher priority, the processor will call the MCE handler, and the SMI# assertion will remain pending. However, while attempting to execute the first instruction of the MCE handler, the SMI# will be recognized and the processor will attempt to execute the SMM handler. If the SMM handler is successfully completed, it will attempt to restart the I/O instruction, but will not have the correct machine state due to the call to the MCE handler. This can lead to failure of the restart and shutdown of the processor.
- If PWRGOOD is de-asserted during a RESET# assertion causing internal glitches, the MCA registers may latch invalid information.
- If RESET# is asserted, then de-asserted, and reasserted, before the processor has cleared the MCA registers, then the information in the MCA registers may not be reliable, regardless of the state or state transitions of PWRGOOD.
- If MCERR# is asserted by one processor and observed by another processor, the observing processor does not log the assertion of MCERR#. The Machine Check Exception (MCE) handler called upon assertion of MCERR# will not have any way to determine the cause of the MCE.
- The Overflow Error bit (bit 62) in the IA32_MC0_STATUS register indicates, when set, that a machine check error occurred while the results of a previous error were still in the error reporting bank (i.e. The Valid bit was set when the new error occurred). If an uncorrectable error is logged in the error-reporting bank and another error occurs, the overflow bit will not be set.
- The MCA Error Code field of the IA32_MC0_STATUS register gets written by a different mechanism than the rest of the register. For uncorrectable errors, the other fields in the IA32_MC0_STATUS register are only updated by the first error. Any further errors that are

detected will update the MCA Error Code field without updating the rest of the register, thereby leaving the IA32_MC0_STATUS register with stale information.

- When a speculative load operation hits the L2 cache and receives a correctable error, the IA32_MC1_Status Register may be updated with incorrect information. The IA32_MC1_Status Register should not be updated for speculative loads.
- The processor should only log the address for L1 parity errors in the IA32_MC1_Status register if a valid address is available. If a valid address is not available, the Address Valid bit in the IA32_MC1_Status register should not be set. In instances where an L1 parity error occurs and the address is not available because the linear to physical address translation is not complete or an internal resource conflict has occurred, the Address Valid bit is incorrectly set.
- The processor may hang when an instruction code fetch receives a hard failure response from the system bus. This occurs because the bus control logic does not return data to the core, leaving the processor empty. IA32_MC0_STATUS MSR does indicate that a hard fail response occurred.
- The processor may hang when the following events occur and the machine check exception is enabled, CR4.MCE=1. A processor that has it's STPCLK# pin asserted will internally enter the Stop Grant State and finally issue a Stop Grant Acknowledge special cycle to the bus. If an uncorrectable error is generated during the Stop Grant process it is possible for the Stop Grant special cycle to be issued to the bus before the processor vectors to the machine check handler. Once the chipset receives its last Stop Grant special cycle it is allowed to ignore any bus activity from the processors. As a result, processor accesses to the machine check handler may not be acknowledged, resulting in a processor hang.

Implication: The processor is unable to correctly report and/or recover from certain errors

Workaround: None identified

Status: For the steppings affected, see the *Summary Tables of Changes*.

P11. Debug Mechanisms May Not Function as Expected

Problem: Certain debug mechanisms may not function as expected on the processor. The cases are as follows:

- When the following conditions occur: 1) An FLD instruction signals a stack overflow or underflow, 2) the FLD instruction splits a page-boundary or a 64 byte cache line boundary, 3) the instruction matches a Debug Register on the high page or cache line respectively, and 4) the FLD has a stack fault and a memory fault on a split access, the processor will only signal the stack fault and the debug exception will not be taken.
- When a data breakpoint is set on the ninth and/or tenth byte(s) of a floating point store using the Extended Real data type, and an unmasked floating point exception occurs on the store, the break point will not be captured.
- When any instruction has multiple debug register matches, and any one of those debug registers is enabled in DR7, all of the matches should be reported in DR6 when the processor goes to the debug handler. This is not true during a REP instruction. As an example, during execution of a REP MOVSW instruction the first iteration a load matches DR0 and DR2 and sets DR6 as FFFF0FF5h. On a subsequent iteration of the instruction, a load matches only DR0. The DR6 register is expected to still contain FFFF0FF5h, but the processor will update DR6 to FFFF0FF1h.
- A Data breakpoint that is set on a load to uncacheable memory may be ignored due to an internal segment register access conflict. In this case the system will continue to execute instructions, bypassing the intended breakpoint. Avoiding having instructions that access segment descriptor

registers e.g. LGDT, LIDT close to the UC load, and avoiding serialized instructions before the UC load will reduce the occurrence of this erratum.

Implication: Certain debug mechanisms do not function as expected on the processor.

Workaround: None identified.

Status: For the steppings affected, see the *Summary Tables of Changes*.

P12. Processor May Live-lock if PDEs or PTEs are in UC Space

Problem: The processor may livelock under the following boundary conditions:

- The Page-Directory Entries (PDEs) or Page-Table Entries (PTEs) are in uncacheable (UC) space
- An instruction fetch misses the ITLB resulting in a page walk
- This instruction fetch is immediately followed by a store that splits a page boundary

Implication: When this erratum occurs, the processor will livelock. This erratum was found using random instruction testing and has not been observed with commercial software.

Workaround: It is possible for the BIOS to contain a workaround for this erratum.

Status: For the steppings affected, see the *Summary Tables of Changes*.

P13. Thermal Status Log Bit May Not be Set when the Thermal Control Circuit is Active

Problem: Bit 1 of the IA32_THERM_STATUS register (Thermal Status Log) is a sticky bit designed to be set to '1' if the thermal control circuit (TCC) has been active since either the previous processor reset or software cleared this bit. If TCC is active and the Thermal Status Log bit is cleared by a processor reset or by software, it will remain clear (set to ‘0’) as long as the TCC remains active. Once TCC deactivates, the next activation of the TCC will set the Thermal Status Log bit.

Implication: When this erratum occurs, the Thermal Status Log bit will be cleared (set to ‘0’) although the thermal control circuit is active.

Workaround: None identified

Status: For the steppings affected, see the *Summary Tables of Changes*.

P14. Processor May Timeout Waiting For A Device To Respond After 0.67 Seconds

Problem: The PCI 2.1 target initial latency specification allows two seconds for a device to respond during initialization-time. The processor may timeout after only approximately 0.67 seconds. When the processor times out it will hang with IERR# asserted. PCI devices that take longer than 0.67 seconds to initialize may not be initialized properly.

Implication: System may hang with IERR# asserted.

Workaround: None identified

Status: For the steppings affected, see the *Summary Tables of Changes*.

P15. Cascading Of Performance Counters Does Not Work Correctly When Forced Overflow is Enabled

Problem: The performance counters are organized into pairs. When the CASCADE bit of the Counter Configuration Control Register (CCCR) is set, a counter that overflows will continue to count in the other counter of the pair. The FORCE_OVF bit forces the counters to overflow on every non-zero increment. When the FORCE_OVF bit is set, the counter overflow bit will be set but the counter no longer cascades.

Implication: The performance counters do not cascade when the FORCE_OVF bit is set.

Workaround: None identified

Status: For the steppings affected, see the *Summary Tables of Changes*.

P16. EMON Event Counting of x87 Loads May Not Work as Expected

Problem: If a performance counter is set to count x87 loads and floating-point exceptions are unmasked, the FPU Operand (Data) Pointer (FDP) may become corrupted.

Implication: When this erratum occurs, FPU Operand (Data) Pointer (FDP) may become corrupted.

Workaround: This erratum will not occur with floating point exceptions masked. If floating-point exceptions are unmasked, then performance counting of x87 loads should be disabled.

Status: For the steppings affected, see the *Summary Tables of Changes*.

P17. Simultaneous Code Breakpoint and Uncorrectable Error Results in Processor Hang

Problem: If an instruction fetch results in an uncorrectable error and there is also a debug breakpoint at this address, the processor will hang and the uncorrectable error will not be logged in the Machine Check registers.

Implication: When this erratum occurs the processor will hang.

Workaround: None identified

Status: For the steppings affected, see the *Summary Tables of Changes*.

P18. Software Controlled Clock Modulation Using A 12.5% or 25% Duty Cycle May Cause The Processor To Hang

Problem: Per the ACPI 1.0b specification, processor clock modulation may be controlled via a processor register (IA32_THERM_CONTROL). The On-Demand Clock Modulation Duty Cycle is controlled by bits 3:1. If these bits are set to a duty cycle of 12.5% or 25%, the processor may hang while attempting to execute a floating-point instruction. In this failure, the last instruction pointer (LIP) is pointing to a floating-point instruction whose instruction bytes are in UC space and which takes an exception 16 (floating point error exception). The processor stalls trying to fetch the bytes of the faulting floating-point instruction and those following it. This processor hang is caused by interactions between the thermal control circuit and floating-point event handler.

Implication: When the clock modulation is set to 12.5% or 25% duty cycle, the processor will go into a sleep state from which it fails to return.

Workaround: Use a duty cycle other than 12.5% or 25%.

Status: For the steppings affected, see the *Summary Tables of Changes*.

P19. RFO With ECC Error May Result In Incorrect Data

Problem: This erratum occurs as the result of the following conditions:

- A read for ownership (RFO) generates a correctable error.
- In the process of correcting the error, a locked RFO (LRFO) is issued that uses the same internal buffer as the previous RFO.
- Another processor issues a snoop to the same address as the LRFO.

An internal boundary condition exists which may prevent the LRFO from completing correctly causing the snoop to receive incorrect data. Intel has not been able to reproduce this erratum with commercial software.

Implication: When this erratum occurs, data corruption may result.

Workaround: None identified

Status: For the steppings affected, see the *Summary Tables of Changes*.

P20. Speculative Page Fault May Cause Livelock

Problem: If the processor detects a page fault, which is corrected before the operating system page fault handler can be called (e.g. a second processor or DMA activity modifies the page tables and the corrected page tables are left in a non-accessed or non-modified state) the processor may livelock. Intel has not been able to reproduce this erratum with commercial software.

Implication: This erratum occurs in systems where page tables are being modified by other processors. If this erratum is encountered, the processor will livelock resulting in a system hang or operating system failure.

Workaround: It is possible for the BIOS to contain a workaround for this erratum.

Status: For the steppings affected, see the *Summary Tables of Changes*.

P21. PAT Index MSB May Be Calculated Incorrectly

Problem: When Mode B or Mode C paging support is enabled and all of the following events occur:

- A page walk returns the Page Directory Entry (PDE) for a large page from memory.
- A subsequent page walk returns the Page Table Entry (PTE) for a 4k page from memory and the Page Attribute Table (PAT) upper index bit in this PTE is set to 1b.

It is possible that the PAT upper index bit in the PTE is incorrectly ignored and assumed to be 0b. The result is that the memory type in the PAT that should have come from the corresponding PAT index [4-7] incorrectly comes from PAT index [0-3].

Implication: If an operating system has programmed the PAT in an asymmetrical fashion i.e. PAT[0-3] is different from PAT[4-7] then an incorrect memory type may be used.

Workaround: None identified

Status: For the steppings affected, see the *Summary Tables of Changes*.

P22. System Bus Interrupt Messages Without Data And Which Receive A HardFailure Response May Hang The Processor

Problem: When a system bus agent (processor or chipset) issues an interrupt transaction without data onto the system bus, and the transaction receives a HardFailure response, a potential processor hang can occur. The processor, which generates an inter-processor interrupt (IPI) that receives HardFailure response, will still log the MCA error event cause as HardFailure, even if the APIC causes a hang. Other processors, which are true targets of the IPI, will also hang on hardfailure-without-data, but will not record an MCA HardFailure event as a cause. If a HardFailure response occurs on a system bus interrupt message with data, the APIC will complete the operation so as not to hang the processor.

Implication: The processor may hang.

Workaround: None identified

Status: For the steppings affected, see the *Summary Tables of Changes*.

P23. SQRTPD and SQRTSD May Return QNaN Indefinite Instead of Negative Zero

Problem: When DAZ mode is enabled, and a SQRTPD or SQRTSD instruction has a negative denormal operand, the instruction will return a QNaN indefinite when the specified response should be zero.

Implication: When this erratum occurs, the instruction will return a QNaN indefinite when a zero is expected.

Workaround: Ensure that negative denormals are not used as operands to the SQRTPD or SQRTSD instructions when DAZ mode is enabled. Software could enable FTZ mode to ensure that negative denormals are not generated by computation prior to execution of the SQRTPD or SQRTSD instructions.

Status: For the steppings affected, see the *Summary Tables of Changes*.

P24. Bus Invalidate Line Requests that Returns Unexpected Data May Result in L1 Cache Corruption

Problem: When a Bus Invalidate Line (BIL) request receives unexpected data from a deferred reply, and a store operation write combines to the same address, there is a small window where the L1 cache is corrupt, and loads can retire with this corrupted data. This erratum occurs in the following scenario:

- A Read-For-Ownership (RFO) transaction is issued by the processor and hits a line in shared state in the L2 cache.
- The RFO is then issued on the system bus as a 0 length Read-Invalidate (BIL), since it doesn't need data, just ownership of the cache line.
- This transaction is deferred by the chipset.
- At some later point, the chipset sends a deferred reply for this transaction with an implicit write-back response. For this erratum to occur, no snoop of this cache line can be issued between the BIL and the deferred reply.
- The processor issues a write-combining store to the same cache line while data is returning to the processor. This store straddles an 8-byte boundary.

Note: Due to an internal boundary condition, a time window exists where the L1 cache contains corrupt data, which could be accessed by a load.

Implication: The L1 cache may contain corrupted data. No known commercially available chipsets trigger the failure conditions.

Workaround: The chipset could issue a BIL (snoop) to the deferred processor to eliminate the failure conditions.

Status: For the steppings affected, see the *Summary Tables of Changes*.

P25. Multi-processor Boot Protocol may not Complete with an IOQ Depth of One

Problem: When the In-Order Queue (IOQ) depth is managed by the chipset to be one entry deep, the system may hang during the multi-processor boot protocol. This hang occurs when the chipset drives BNR# in such a way that the processors are continually throttled off the bus then released to access the bus in alternating cycles which never allows the multi-processor boot protocol to complete execution.

Implication: The system may hang during the multi-processor boot protocol.

Workaround: If the chipset drives BNR# in such a way that the processors are continually throttled off the bus then released to access the bus in alternating cycles, do not use In-Order Queue de-pipelining.

Status: For the steppings affected, see the *Summary Tables of Changes*.

P26. Processor Flags #PF Instead of #AC on an Unlocked CMPXC8B Instruction

Problem: If a data page fault (#PF) and alignment check fault (#AC) both occur for an unlocked CMPXC8B instruction, then #PF will be flagged.

Implication: Software that depends #AC before #PF will be affected since #PF is flagged in this case.

Workaround: Remove the software's dependency on the fact that #AC has precedence over #PF. Alternately, if the reload is due to a not present page, reload the page in the page fault handler and then restart the faulting instruction.

Status: For the steppings affected, see the *Summary Tables of Changes*.

P27. Incorrect Data may be Returned When Page Tables are Located in Write Combining (WC) Memory

Problem: If page directories and/or page tables are located in Write Combining (WC) memory, speculative loads to cacheable memory may complete with incorrect data.

Implication: Cacheable loads to memory mapped using page tables located in write combining memory may return incorrect data. Intel has not been able to reproduce this erratum with commercially available software.

Workaround: Do not place page directories and/or page tables in WC memory.

Status: For the steppings affected, see the *Summary Tables of Changes*.

P28. FSW May not be Completely Restored after Page Fault on FRSTOR or FLDENV Instructions

Problem: If the FPU operating environment or FPU state (operating environment and register stack) being loaded by an FLDENV or FRSTOR instruction wraps around a 64-Kbyte or 4-Gbyte boundary and a page fault (#PF) or segment limit fault (#GP or #SS) occurs on the instruction near the wrap boundary, the upper byte of the FPU status word (FSW) might not be restored. If the fault handler does not restart program execution at the faulting instruction, stale data may exist in the FSW.

Implication: When this erratum occurs, stale data will exist in the FSW.

Workaround: Ensure that the FPU operating environment and FPU state do not cross 64-Kbyte or 4-Gbyte boundaries. Alternately, ensure that the page fault handler restarts program execution at the faulting instruction after correcting the paging problem.

Status: For the steppings affected, see the *Summary Tables of Changes*.

P29. Write Combining (WC) Load May Result in an Unintended Address on System Bus

Problem: When the processor performs a speculative write combining (WC) load, down the path of a mispredicted branch, and the address happens to match a valid UnCacheable (UC) address translation with the Data Translation Look-Aside Buffer, an unintended UnCacheable load operation may be sent out on the system bus.

Implication: When this erratum occurs, an unintended load may be sent on system bus. Intel has only encountered this erratum during pre-silicon simulation.

Workaround: It is possible for the BIOS to contain a workaround for this erratum for some steppings of the processor.

Status: For the steppings affected, see the *Summary Tables of Changes*.

P30. Processor Issues Inconsistent Transaction Size Attributes for Locked Operation

Problem: When the processor is in the Page Address Extension (PAE) mode and detects the need to set the Access and/or Dirty bits in the page directory or page table entries, the processor sends an 8 byte load lock onto the system bus. A subsequent 8 byte store unlock is expected, but instead a 4 byte store unlock occurs. Correct data is provided since only the lower bytes change, however external logic monitoring the data transfer may be expecting an 8-byte store unlock.

Implication: No known commercially available chipsets are affected by this erratum.

Workaround: None identified at this time.

Status: For the steppings affected, see the *Summary Tables of Changes*.

P31. Multiple Accesses to the Same S-State L2 Cache Line and ECC Error Combination May Result in Loss of Cache Coherency

Problem: When a Read For Ownership (RFO) cycle has a 64 bit address match with an outstanding read hit on a line in the L2 cache which is in the S-state AND that line contains an ECC error, the processor should recycle the RFO until the ECC error is handled. Due to this erratum, the processor does not recycle the RFO and attempt to service both the RFO and the read hit at the same time.

Implication: When this erratum occurs, cache may become incoherent.

Workaround: None identified at this time.

Status: For the steppings affected, see the *Summary Tables of Changes*.

P32. IA32_MC0_ADDR and IA32_MC0_MISC Registers Will Contain Invalid or Stale Data Following a Data, Address, or Response Parity Error

Problem: If the processor experiences a data, address, or response parity error, the ADDRV and MISCV bits of the IA32_MC0_STATUS register are set, but the IA32_MC0_ADDR and IA32_MC0_MISC registers are not loaded with data regarding the error.

Implication: When this erratum occurs, the IA32_MC0_ADDR and IA32_MC0_MISC registers will contain invalid or stale data.

Workaround: Ignore any information in the IA32_MC0_ADDR and IA32_MC0_MISC registers after a data or response parity error.

Status: For the steppings affected, see the *Summary Tables of Changes*.

P33. When the Processor is in the System Management Mode (SMM), Debug Registers may be Fully Writeable

Problem: When in System Management Mode (SMM), the processor executes code and stores data in the SMRAM space. When the processor is in this mode and writes are made to DR6 and DR7, the processor should block writes to the reserved bit locations. Due to this erratum, the processor may not block these writes. This may result in invalid data in the reserved bit locations.

Implication: Reserved bit locations within DR6 and DR7 may become invalid.

Workaround: Software may perform a read/modify/write when writing to DR6 and DR7 to ensure that the value in the reserved bits are maintained.

Status: For the steppings affected, see the *Summary Tables of Changes*.

P34. Associated Counting Logic must be Configured when Using Event Selection Control (ESCR) MSR

Problem: ESCR MSRs allow software to select specific events to be counted, with each ESCR usually associated with a pair of performance counters. ESCRs may also be used to qualify the detection of at-retirement events that support precise-event-based sampling (PEBS). A number of performance metrics that support PEBS require a 2nd ESCR to tag uops for the qualification of at-retirement events. (The first ESCR is required to program the at-retirement event.) Counting is enabled via counter configuration control registers (CCCR) while the event count is read from one of the associated counters. When counting logic is configured for the subset of at-retirement events that require a 2nd ESCR to tag uops, at least one of the CCCRs in the same group of the 2nd ESCR must be enabled.

Implication: If no CCCR/counter is enabled in a given group, the ESCR in that group that is programmed for tagging uops will have no effect. Hence a subset of performance metrics that require a 2nd ESCR for tagging uops may result in 0 count.

Workaround: Ensure that at least one CCCR/counter in the same group as the tagging ESCR is enabled for those performance metrics that require 2 ESCRs and tagging uops for at-retirement counting.

Status: For the steppings affected, see the *Summary Tables of Changes*.

P35. Livelock May Occur When Bus Parking Is Disabled

Problem: A livelock may occur when processor bus parking is disabled, and when (1) the processor is the symmetric owner of the bus with one internal request pending, and (2) the processor observes the assertion of BPRI#, BNR# or a full In Order Queue (IOQ). In this scenario, the processor bus interface unit assumes that the assertion of ADS# is not required, deasserts BREQ, and, as a result, relinquishes bus ownership without issuing the pending request. If the BPRI#, BNR# or full IOQ pattern continues coincident with the arbitration phase of the processor that still has only one outstanding internal request, livelock may occur. Assertion of bus parking, any change to the regular pattern of BPRI# or BNR# assertion noted above, or the arrival of a second internal transaction will release the processor from the livelock condition.

Implication: This erratum may result in a livelock.

Workaround: This erratum can be avoided by enabling bus parking. The deassertion of signal A15# during the active-to-inactive edge of RESET# will enable bus parking.

Status: For the stepping affected, see the *Summary of Changes* at the beginning of this section.

.

P36. CR2 May Be Incorrect or an Incorrect Page Fault Error Code May Be Pushed onto Stack After Execution of an LSS Instruction

Problem: Under certain timing conditions, the internal load of the selector portion of the LSS instruction may complete with potentially incorrect speculative data before the load of the offset portion of the address completes. The incorrect data is corrected before the completion of the LSS instruction but the value of CR2 and the error code pushed on the stack are reflective of the speculative state. Intel has not observed this erratum with commercially available software.

Implication: When this erratum occurs, the contents of CR2 may be off by two, or an incorrect page fault error code may be pushed onto the stack.

Workaround: It is possible for the BIOS to contain a workaround for this erratum.

Status: For the steppings affected, see the *Summary Tables of Changes*.

P37. Buffer on Resistance May Exceed Specification

Problem: The datasheet specifies the resistance range for RON (Buffer On Resistance) for the AGTL+ buffer as 5 to 11 ohms. Due to this erratum, RON may be as high as 12 ohms.

Implication: The RON value affects the voltage level of the signals when the buffer is driving the signal low. A higher RON may adversely affect the system's ability to meet specifications such as VIL. As the system design also affects margin to specification, designs may or may not have sufficient margin to function properly with an increased RON. System designers should evaluate whether a particular system is affected by this erratum. Designs that follow the recommendations in the Intel® Xeon™ Processor and Intel® 860 Chipset Platform Design Guide are not expected to be affected.

Workaround: No workaround is necessary for systems with margin sufficient to accept a higher R_{ON}.

Status: For the steppings affected, see the *Summary Tables of Changes*.

P38. Instruction Pointer Stored on Stack May Become Invalid

Problem: The instruction pointer stored on the stack my become invalid due to an internal boundary condition which may exist on a Hyper Threading Technology enabled processors. The following sequence of events must occur in order to encounter this erratum:

1. One logical processor executes the WRMSR instruction with incorrect data causing a general protection fault
2. Simultaneously, an event that requires micro-architectural synchronization among the two logical processors occurs on the second logical processor. This event may cause an invalid instruction pointer to be stored on the ring 0 stack during the transition to GP fault handler on the first logical processor.

Implication: The instruction pointer stored on the stack may be invalid, potentially causing errors during execution of or return from the GP fault handler.

Workaround: It is possible for the BIOS to contain a workaround for this erratum.

Status: For the steppings affected, see the *Summary Tables of Changes*.

P39. Shutdown and IERR# May Result Due to a Machine Check Exception on a Hyper-Threading Technology Enabled Processor

Problem: When a Machine Check Exception (MCE) occurs due to an internal error, both logical processors on a Hyper Threading technology enabled processor normally vector to the MCE handler. However, if one of the logical processors is in the "Wait for SIPI" state, that logical processor will not have a MCE handler and will shut down and assert IERR#.

Implication: A processor with a logical processor in the "Wait for SIPI" state will shut down when an MCE occurs on the other thread.

Workaround: None identified at this time.

Status: For the steppings affected, see the *Summary Tables of Changes*.

P40. Hyper-Threading Technology Enabled Processors May Hang in the Presence of Extensive Self-Modifying Code

Problem: For multi-processor platforms, in which Hyper-Threading Technology enabled processors are executing extensive self modifying code, and branch trace messages are enabled on at least one logical processor, the system may hang. In this scenario, a processor executing within 1K of code being written to by another processor may attempt to end this flow, thereby resulting in a hang.

Implication: When this erratum occurs the system will hang.

Workaround: It is possible for the BIOS to contain a workaround for this erratum.

Status: For the steppings affected, see the *Summary Tables of Changes*.

P41. Global Bit Incorrectly Set for Secondary Logical Processors in ITLB

Problem: Due to a boundary condition in the translation look-aside buffer logic, the global bit information in the TLB entry for a mapping belonging to the first logical processor can overwrite the global bit information for a mapping belonging to the second logical processor. This occurs in the following scenario:

- The first logical processor misses the ITLB resulting in a page walk
- The second logical processor also misses the ITLB and generates a page walk.

Note: In certain timing scenarios within the processor, the leftover global bit information from the first logical processor may overwrite the second logical processor.

Implication: When this erratum occurs, if the Page global bit for the second logical processor is overwritten with a 0b, this will result in performance degradation for the first logical processor. If the page global bit is incorrectly changed from a 0 to 1, this erratum may result in software failures.

Workaround: It is possible for BIOS code to contain a workaround for this erratum.

Status: For the steppings affected, see the *Summary Tables of Changes*.

P42. Machine Check Exception (MCE) Observed on DP Platforms

Problem: A system bus address parity error may be signaled if two processors run at odd core frequency to system bus-ratios (17:1, 19:1, etc) on DP processor platforms. This address parity error signaling issue does not occur if the processors run at even bus-ratios.

Implication: A Machine Check Exception (MCE) may be observed on DP platforms.

Workaround: The system BIOS should ensure that processors run at even core frequency to system bus-ratios (16:1, 18:1, etc).

Status: For the steppings affected, see the *Summary Tables of Changes*.

P43. BPM[5:3]# VIL Does Not Meet Specification

Problem: The VIL for BPM[5:3]# is specified as 0.9 * GTLREF [V]. Due to this erratum the VIL for these signals is 0.9 * GTLREF - .100 [V].

Implication: The processor requires a lower input voltage than specified to recognize a low voltage on the BPM[5:3]# signals.

Workaround: When intending to drive the BPM[5:3]# signals low, ensure that the system provides a voltage lower than 0.9 * GTLREF - .100 [V].

Status: For the steppings affected, see the *Summary Tables of Changes*.

P44. Processor May Hang Under Certain Frequencies and 12.5% STPCLK# Duty Cycle

Problem: If a system de-asserts STPCLK# at a 12.5% duty cycle, and the processor is running below 2 GHz, and the processor thermal control circuit (TCC) on-demand clock modulation is active, the processor may hang. This erratum does not occur under the automatic mode of the TCC.

Implication: When this erratum occurs, the processor will hang.

Workaround: If use of the on-demand mode of the processor's TCC is desired in conjunction with STPCLK# modulation, then assure that STPCLK# is not asserted at a 12.5% duty cycle.

Status: For the steppings affected, see the *Summary Tables of Changes*.

P45. System May Hang if a Fatal Cache Error Causes Bus Write Line (BWL) Transaction to Occur to the Same Cache Line Address as an Outstanding Bus Read Line (BRL) or Bus Read-Invalidate Line (BRIL)

Problem: A processor internal cache fatal data ECC error may cause the processor to issue a BWL transaction to the same cache line address as an outstanding BRL or BRIL. As it is not typical behavior for a single processor to have a BWL and a BRL/BRIL concurrently outstanding to the same address, this may represent an unexpected scenario to system logic within the chipset.

Implication: The processor may not be able to fully execute the machine check handler in response to the fatal cache error if system logic does not ensure forward progress on the system bus under this scenario.

Workaround: System logic should ensure completion of the outstanding transactions. Note that during recovery from a fatal data ECC error, memory image coherency of the BWL with respect to BRL/BRIL transactions is not important. Forward progress is the primary requirement.

Status: For the steppings affected, see the *Summary Tables of Changes*.

P46. L2 Cache May Contain Stale Data in the Exclusive State

Problem: If a cacheline (A) is in Modified (M) state in the write-combining (WC) buffers and in the Invalid (I) state in the L1 cache and it's adjacent sector (B) is in the Invalid (I) state and the following scenario occurs:

1. A read to B misses in the L2 cache and allocates cacheline B and its associated second-sector pre-fetch into an almost full bus queue,
2. A Bus Read Line (BRL) to cacheline B completes with HIT# and fills data in Shared (S) state,
3. The bus queue full condition causes the prefetch to cacheline A to be cancelled, cacheline A will remain M in the WC buffers and I in the L2 while cacheline B will be in the S state. Then, if the further conditions occur:
    1. Cacheline A is evicted from the WC Buffers to the bus queue which is still almost full,
    2. A hardware prefetch Read for Ownership (RFO) to cacheline B, hits the S state in the L2 and observes cacheline A in the I state, allocates both cachelines,

3. An RFO to cacheline A completes before the WC Buffers write modified data back, filling the L2 with stale data,

4. The write-back from the WC Buffers completes leaving stale data, for cacheline A, in the Exclusive (E) state in the L2 cache.

Implication: Stale data may be consumed leading to unpredictable program execution. Intel has not been able to reproduce this erratum in commercial software.

Workaround: It is possible for BIOS to contain a workaround for this erratum.

Status: For the steppings affected, see the *Summary Tables of Changes*.

P47. Re-mapping the APIC Base Address to a Value Less Than or Equal to 0xDC001000 may Cause IO and Special Cycle Failure

Problem: Re-mapping the APIC base address from its default can cause conflicts with either I/O or special cycle bus transactions.

Implication: Either I/O or special cycle bus transactions can be redirected to the APIC, instead of appearing on the front-side bus.

Workaround: Use any APIC base addresses above 0xDC001000 as the relocation address.

Status: For the steppings affected, see the *Summary Tables of Changes*.

P48. Erroneous BIST Result Found in EAX Register After Reset

Problem: The processor may show an erroneous BIST (built-in self test) result in the EAX register bit 0 after reset.

Implication: When this erratum occurs, an erroneous BIST failure will be reported in the EAX register bit 0, however this failure can be ignored since it is not accurate.

Workaround: It is possible for BIOS to workaround this issue by masking off bit 0 in the EAX register where BIST results are written.

Status: For the steppings affected, see the *Summary Tables of Changes*.

P49. Processor Does Not Flag #GP on Non-Zero Write to Certain MSRs

Problem: When a non-zero write occurs to the upper 32 bits of IA32_CR_SYSENTER_EIP or IA32_CR_SYSENTER_ESP, the processor should indicate a general protection fault by flagging #GP. Due to this erratum, the processor does not flag #GP.

Implication: The processor unexpectedly does not flag #GP on a non-zero write to the upper 32 bits of IA32_CR_SYSENTER_EIP or IA32_CR_SYSENTER_ESP. No known commercially available operating system has been identified to be affected by this erratum.

Workaround: None identified.

Status: For the steppings affected, see the *Summary Tables of Changes*.

P50. Simultaneous Assertion of A20M# and INIT# May Result in Incorrect Data Fetch

Problem: If A20M# and INIT# are simultaneously asserted by software, followed by a data access to the 0xFFFFFXXX memory region, with A20M# still asserted, incorrect data will be accessed. With A20M# asserted, an access to 0xFFFFFXXX should result in a load from physical address 0xFFEFFXXX. However, in the case of A20M# and INIT# being asserted together, the data load will actually be from the physical address 0xFFFFFXXX. Code accesses are not affected by this erratum.

Implication: Processor may fetch incorrect data, resulting in BIOS failure.

Workaround: Deasserting and reasserting A20M# prior to the data access will workaround this erratum.

Status: For the steppings affected, see the *Summary Tables of Changes*.

P51. Processor Does not Respond to Break Requests From ITP

Problem: On power-up and low-power state transitions, the processor's TAP circuitry may remain in the Tap-Logic-Reset (TLR) state

Workaround: None identified

Implication: The ITP is unable to cause a break on reset in the processor, which may prevent the loading of processor and chipset registers, or affect the ability to debug from cold boot and low power transitions

Status: For the stepping affected, see the *Summary of Changes* at the beginning of this section.

P52. Glitches on Address and Data Strobe Signals May Cause System Shutdown

Problem: When the CPUID instruction is executed with EAX = 2 on a processor without Hyper-Threading Technology or with Hyper-Threading Technology disabled via power on configuration, it should return a value of 51h in EAX[15:8] to indicate that the Instruction Translation Lookaside Buffer (ITLB) has 128 entries. On a processor with Hyper-Threading Technology enabled, the processor should return 50h (64 entries). Due to this erratum, the CPUID instruction always returns 50h (64 entries).

Implication: Software may incorrectly report the number of ITLB entires. Operation of the processor is not affected.

Workaround: None identified.

Status: For the steppings affected see the Summary of Changes at the beginning of this section.

Specification Changes

P3. Direction Flag (DF) Mistakenly Denoted as a System Flag

The Intel Architecture Software Developer's Manual, Vol 1: Basic Architecture Section 3.4.3 "EFLAGS Register", in Figure 3-7 EFLAGS Register currently states:

X Direction Flag(DF)

It should state:

C Direction Flag(DF)

P4. Fopcode Compatibility Mode

The Intel Architecture Software Developer's Manual, Vol 1: Basic Architecture Section 8.1.8.1 "FOPCODE COMPATIBILITY MODE" currently states:

"When the FOP code compatibility mode is enabled, the IA32 architecture guarantees that if an unmasked x87 FPU floating-point exception is generated, the opcode of the last non-control instruction executed prior to the generation of the exception will be stored in the x87 FPU opcode register, and that value can be read by a subsequent FSAVE of FXSAVE instruction. When the fop compatibility mode is disabled (default), the value stored in the x87 FPU opcode register is undefined (reserved)."

It should state:

"If FOP code compatibility mode is enabled, the FOP is defined as it
has always been in previous IA32 implementations (always defined as the FOP
of the last non-transparent FP instruction executed before a FSAVE/FSTENV/FXSAVE).

If FOP code compatibility mode is disabled (default), FOP is only valid if
the last non-transparent FP instruction executed before a
FSAVE/FSTENV/FXSAVE had an unmasked exception."

P5. FCOS, FPTAN, FSIN, and FSINCOS Trigonometric Domain not Correct

The Intel Architecture Software Developer's Manual, Vol 2: Instruction Set Reference Section 3.2 "INSTRUCTION REFERENCE" FCOS, FPTAN, FSIN, and FSINCOS trigonometric domain for C2 is incorrect. Under the FPU Flags affected, C2 currently states:

C2 Set to 1 if source operand is outside the range -263 to +263; otherwise, cleared to 0.

It should state:

C2 Set to 1 if outside range -263 <source operand <+263; otherwise, set to 0.

P6. Incorrect Description of Stack

The Intel Architecture Software Developer's Manual, Volume 1: Basic Architecture Chapter 6, Section 6.2 paragraph 2, labeled "STACK" currently states:

The next available memory location on the stack is called the top of stack. At any given time, the stack pointer (contained in the ESP register) gives the address (that is the offset from the base of the SS segment) of the top of the stack.

This paragraph is incorrect and will be removed from the section listed above.

P7. EFLAGS Register Correction

The Intel Architecture Software Developer's Manual, Volume 1: Basic Architecture Section 3.7.2, Figure 3.7 "EFLAGS Register," currently states:

Bit 11 "OF" as "X"

It should state:

Bit 11 "OF" as "S"

P8. PSE-36 Paging Mechanism

The Intel Architecture Software Developer's Manual, Vol 3: System Programming Guide Chapter 3, Section 3.9, third paragraph currently states:

As is shown in Table 3-3, the following flags must be set or cleared to enable the PSE-36 paging mechanism:

PSE-36 CPUID feature flag-When set, it indicates the availability of the PSE-36 paging mechanism on the IA-32 processor on which the CPUID instruction is executed.

PG flag (bit 31) in register CR0-Set to 1 to enable paging.

PSE flag (bit 4) in control register CR4 - Set to 1 to enable the page size extension for 4-Mbyte pages.

PAE flag (bit 5) in control register CR4-Clear to 0 to disable the PAE paging mechanism.

It should state:

As is shown in Table 3-3, the following flags must be set or cleared to enable the PSE-36 paging mechanism:

PSE-36 CPUID feature flag-When set, it indicates the availability of the PSE-36 paging mechanism on the IA-32 processor on which the CPUID instruction is executed.

PG flag (bit 31) in register CR0-Set to 1 to enable paging.

PAE flag (bit 5) in control register CR4-Clear to 0 to disable the PAE paging mechanism.

PSE flag (bit 4) in control register CR4 and the PS flag in PDE- Set to 1 to enable the page size extension for 4-Mbyte pages.

Or the PSE flag (bit 4) in control register CR4- Set to 1 and the PS flag (bit 7) in PDE- Set to 0 to enable 4-KByte pages with 32-bit addressing (below 4GBytes).

P9. 0x33 Opcode

The Intel Architecture Software Developer's Manual, Vol 2: Instruction Set Reference Appendix A, Table A-2 the opcode corresponding to 0x33 currently states:

Gb, Ev

It should state:

Gv, Ev

Also, Page 3-791, XOR-Logical Exclusive OR, the two entries for opcode 33 currently states:

Opcode	Instruction	Description

33 /r	XOR r16,r/m16	r8 XOR r/m8
33 /r	XOR r32,r/m32	r8 XOR r/m8

It should state:

Opcode	Instruction	Description
33 /r	r16 XOR r/m16	r8 XOR r/m8
33 /r	r32 XOR r/m32	r8 XOR r/m8

P10. Incorrect Information for SLDT

In the Intel Architecture Software Developer's Manual, Vol 2: Instruction Set Reference, the opcode/Instruction/Description table for SLDT currently states “SLDT r/m32 Store segment selector from LDTR in low-order 16 bits of r/m32” but should instead state “SLDT r32 Store segment selector from LDTR in low-order 16 bits of r32.”

P11. LGDT/LIDT Instruction Information Correction

In the Intel Architecture Software Developer's Manual, Vol 2: Instruction Set Reference, the sentence in the LGDT/LIDT instruction section currently states:

”See 'SFENCE -- Store Fence' in this chapter for information on storing the contents of the GDTR and IDTR."

It should state:

"See 'SGDT/SIDT' in this chapter for information on storing the contents of the GDTR and IDTR."

P12. Errors in Instruction Set Reference

The following changes will be made to the Intel Architecture Software Developer's Manual, Vol 2: Instruction Set Reference:

Page 3-586 “PMULUDQ—Multiply Packed Unsigned Doubleword Integers”

currently states:

66 OF F4 /r PMULUDQ xmm1, xmm2/m128

It should state:

66 0F F4 /r PMULUDQ xmm1, xmm2/m128

2. Page A-9, Table A-3, Two-byte Opcode Map:08H-7FH (First Byte is 0FH), entry 2B currently states:

MOVNTPS

Wps, Vps

MOVNTPS (66)

Wpd, Vpd

It should state:

MOVNTPS

Wps, Vps

MOVNTPD (66)

Wpd, Vpd

3. Page A-9, Table A-3, Two-byte Opcode Map:08H-7FH (First Byte is 0FH).

Entry 3C currently states:

Blank (empty space)

It should state:

MOVNTI

4. Page A-10, Table A-3, Two-byte Opcode Map:80H-7FH (First Byte is 0FH).

Entry D7 currently states:

PMOVMSKB

Gd, Pq

PMOVMKSB (66)

Gd, Vdq

It should state:

PMOVMSKB

Gd, Pq

PMOVMSKB (66)

Gd, Vdq

5. Page A-10, Table A-3, Two-byte Opcode Map:80H-7FH (First Byte is 0FH).

Entry F7 currently states:

MASKMOVQ

Ppi, Qpi

MASKMOVQU (66)

Vdq, Wdq

It should state:

MASKMOVQ

Ppi, Qpi

MASKMOVDQU (66)

Vdq, Wdq

6. Page A-11, Table A-3, Two-byte Opcode Map:88H-7FH (First Byte is 0FH).

The title table currently states:

Table A-3. Two-byte Opcode Map:88H-7FH (First Byte is FFH)

It should state:

Table A-3. Two-byte Opcode Map:88H-7FH (First Byte is 0FH)

7. Page A-11, Table A-3, Two-byte Opcode Map:88H-7FH (First Byte is 0FH).

Entry FB currently states:

PSUBD

Pq, Qq

PSUBD (66)

Vdq, Wdq

It should state:

PSUBQ

Pq, Qq

PSUBQ (66)

Vdq, Wdq

8. Page B-21, Table B-12, MMX Instruction Formats and Encodings (Contd.).

Entry PMADD currently states:

PMADD – Packed Multiply add

It should state:

PMADDWD – Packed Multiply add

9. Page B-21, Table B-12, MMX Instruction Formats and Encodings (Contd.).

Entry PMULH currently states:

PMULH – Packed multiplication

It should state:

PMULHW – Packed multiplication, store high word

10. Page B-21, Table B-12, MMX Instruction Formats and Encodings (Contd.).

Add instruction PMULHUW :

PMULHUW – Packed multiplication, store high word (unsigned)

mmxreg2 to mmxreg1	0000 1111: 1110 0100: 11 mmxreg1 mmxreg2
memory to mmxreg	0000 1111: 1110 0100: mod mmxreg r/m

11. Page B-21, Table B-12, MMX Instruction Formats and Encodings (Contd.).

Entry PMULL currently states:

PMULL – Packed multiplication

It should state:

PMULLW – Packed multiplication, store low word

12. Page B-40, Table B-19, Formats and Encodings of the SSE2 SIMD Integer Instruction.

Entry PMADD currently states:

PMADD – Packed multiply add

It should state:

PMADDWD – Packed multiply add

13. Page B-41, Table B-19, Formats and Encodings of the SSE2 SIMD Integer Instruction.

Entry PMULH currently states:

PMULH – Packed multiplication

It should state:

PMULHW – Packed multiplication, store high word

14. Page B-41, Table B-19, Formats and Encodings of the SSE2 SIMD Integer Instruction.

Add instruction PMULHUW:

PMULHUW – Packed multiplication, store high word (unsigned)

xmmreg2 to xmmreg1	0110 0110 : 0000 1111 : 11110 0100 : 11 xmmreg1 xmmreg2
memory to xmmreg	0110 0110 : 0000 1111 : 1110 0100 : mod xmmreg r/m

15. Page B-41, Table B-19, Formats and Encodings of the SSE2 SIMD Integer Instruction.

Entry PMULL currently states:

PMULL – Packed multiplication

It should state:

PMULLW – Packed multiplication, store low word

P13. RSM Instruction Set Summary

The Intel Architecture Software Developer's Manual, Vol 1: Basic Architecture Section 5.8 "INSTRUCTION SET SUMMARY" currently states:

RSM Return from system management mode (SSM)

It should state:

RSM Return from system management mode (SMM)

P14. Correct MOVAPS and MOVAPD Operand Section

The Intel Architecture Software Developer's Manual, Vol 2: Instruction Set Reference Section 3.2 "INSTRUCTION REFERENCE" MOVAPS and MOVAPD operation section currently states:

Operation

DEST ← SRC;

It should state:

Operation

DEST ← SRC;

* #GP if SRC or DEST unaligned memory operand *;

P15. DAA—Decimal Adjust AL after Addition

The Intel Architecture Software Developer's Manual, Vol 2: Instruction Set Reference, page 3-173

currently states:

Operation

```
IF (((AL AND 0FH) > 9) or AF = 1)
  THEN
    AL ←AL + 6;
    CF ←CF OR CarryFromLastAddition; (* CF OR carry from AL ←AL + 6 *)
    AF ←1;
  ELSE
    AF ←0;
FI;
IF ((AL AND F0H) > 90H) OR CF = 1)
  THEN
    AL ←AL + 60H;
    CF ←1;
  ELSE
    CF ←0;
FI;
```

It should state:

Operation

```
old_AL ← AL;
old_CF ← CF;
CF ← 0;
IF (((AL AND 0FH) > 9) or AF = 1)
  THEN
    AL ←AL + 6;
    CF ←old_CF OR (Carry from AL ←AL + 6);
    AF ←1;
  ELSE
```

```
    AF ←0;
FI;
IF ((old_AL > 99H) OR (old_CF = 1)
  THEN
    AL ←AL + 60H;
    CF ←1;
  ELSE
    CF ←0;
FI;
```

P16. DAS—Decimal Adjust AL after Subtraction

The Intel Architecture Software Developer's Manual, Vol 2: Instruction Set Reference, page 3-175 currently states:

```
Operation
IF (AL AND 0FH) > 9 OR AF = 1
  THEN
    AL ←AL - 6;
    CF ←CF OR CarryFromLastAddition; (* CF OR carry from AL ←AL - 6 *)
    AF ←1;
  ELSE AF ←0;
FI;
IF ((AL > 9FH) OR CF = 1)
  THEN
    AL ←AL - 60H;
    CF ←1;
  ELSE CF ←0;
FI;
```

It should state:

Operation

old_AL ← AL;

old_CF ← CF;

CF ← 0;

IF (((AL AND 0FH) > 9) OR AF = 1)

  THEN

    AL ←AL - 6;

    CF ←old_CF OR (Borrow from AL ←AL - 6);

    AF ←1;

  ELSE

    AF ←0;

FI;

IF ((old_AL > 99H) OR (old_CF = 1))

  THEN

    AL ←AL - 60H;

    CF ←1;

  ELSE

    CF ←0;

FI;

P17. Omission of Dependency between BTM and LBR

The Intel Architecture Software Developer's Manual, Vol 3: System Programming Guide Chapter 15, Section 5.3, on page 15-15 currently states:

15.5.3. Monitoring Branches, Exceptions, and Interrupts (Pentium

4 and Intel Xeon Processors)

When the LBR flag in the IA32_DEBUGCTL MSR is set, the processor automatically begins

recording branch records for taken branches, interrupts, and exceptions (except for debug exceptions)

in the LBR stack MSRs.

When the processor generates a debug exception (#DB), it automatically clears the LBR flag before executing the exception handler, but does not touch the LBR stack MSRs. The branch records for the last four taken branches, interrupts, and/or exceptions are thus retained for anal-ysis by the debugger program.

The debugger can use the linear addresses in the LBR stack to reset breakpoints in the break-point-address registers (DR0 through DR3), allowing a backward trace from the manifestation of a particular bug toward its source.

Before resuming program execution from a debug-exception handler, the handler must set the LBR flag again to re-enable last branch recording.

It should state:

15.5.3. Monitoring Branches, Exceptions, and Interrupts (Pentium 4 and Intel Xeon Processors)
When the LBR flag in the IA32_DEBUGCTL MSR is set, the processor automatically begins recording branch records for taken branches, interrupts, and exceptions (except for debug exceptions) in the LBR stack MSRs.

When the processsor generates a debug exception (#DB), it automatically clears the LBR flag before executing the exception handler. This action does not clear previously stored LBR stack MSRs. The branch record for the last four taken branches, interrupts and/or exceptions are retained for analysis.

A debugger can use the linear addresses in the LBR stack to reset breakpoints in the break-point address registers (DR0 through DR3). This allows a backward trace from the manifestation of a particular bug toward its source.

If the LBR flag is cleared and TR flag in the IA32_DEBUGCTLTR MSR remains set, the processor will continue to update LBR stack MSRs. This is because BTM information must be generated from entries in the LBR stack (see 14.5.5). A #DB does not automatically clear the TR flag.

P18. I/O Permissions Bitmap Base Addy > 0xDFFF Does not Cause #GP(0) Fault

The Intel Architecture Software Developer's Manual, Vol 1: Basic Architecture, page 12-6, section 12.5.2, last paragraph currently states:

If the I/O bit map base address is greater than or equal to the TSS segment limit, there is no I/O permission map, and all I/O instructions generate exceptions when the CPL is greater than the current IOPL. The I/O bit map base address must be less than or equal to DFFFH.

It should state:

If the I/O bit map base address is greater than or equal to the TSS segment limit, there is no I/O permission map, and all I/O instructions generate exceptions when the CPL is greater than the current IOPL.

P19. Wrong Field Width for MINSS and MAXSS

The Intel Architecture Software Developer's Manual, Vol 2: Instruction Set Reference, Section 3.2 Instruction Reference under "MAXSS—Return Maximum Scalar Single-Precision Floating-Point Value" page 3-415 currently states:

DEST[63-0] .IF ((DEST[31-0] = 0.0) AND (SRC[31-0] = 0.0)) THEN SRC[31-0]

It should state:

DEST[31-0] .IF ((DEST[31-0] = 0.0) AND (SRC[31-0] = 0.0)) THEN SRC[31-0]

The Intel Architecture Software Developer's Manual, Vol 2 Instruction Set Reference, Section 3.2 under "MINSS—Return Minimum Scalar Single-Precision floating-Point Value" page 3-428 currently states:

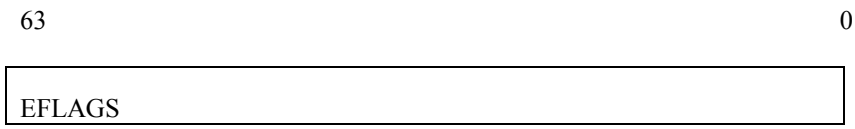
DEST[63-0] .IF ((DEST[31-0] = 0.0) AND (SRC[31-0] = 0.0)) THEN SRC[31-0]

It should state:

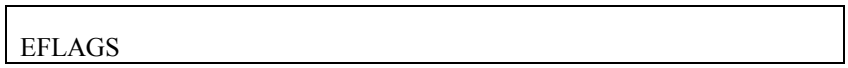
DEST[31-0] .IF ((DEST[31-0] = 0.0) AND (SRC[31-0] = 0.0)) THEN SRC[31-0]

P20. Figure 15-12 PEBS Record Format

The Intel Architecture Software Developer's Manual, Vol 3 System Programming Guide Section 15.9.6 " Programming the Performance Counters for Non-Retirement Events", page 15 - 37, Figure 15-12, first row currently states:



It should state:



P21. I/O Permission Bit Map

The Intel Architecture Software Developer's Manual, Vol. 1: Basic Architecture, Chapter 12, section 12.5.2 on Figure 12-2 (I/O Permission Bit Map) currently states:

Last byte of bit map must be followed by a byte with all bits.

It should state:

Last byte of bit map must be followed by a byte with all bits set.

Also, in the lower left hand Conner of Figure 12-2 (I/O Permission Bit Map) currently states:

Last I/O base map must be

It should state:

Last I/O base map must be less than or equal to DFFFH

P22. Cache Description

The Intel Architecture Software Developer's Manual, Vol 2: Instruction Set Reference, Table 3-10, the "sectored, 64 byte line size" description is used for the following descriptors: 0x22, 0x23, 0x79, 0x7a, 0x7b, 0x7c. This description will change to "dual-sectored line, 64 byte sector size" for clarity.

P23. Instruction Formats and Encoding

The Intel Architecture Software Developer's Manual, Vol 2: Instruction Set Reference, Page B-8, CMOVcc memory to register should be encoded as "0000 1111 : 0100 tttn : mod reg r/m". Page B-8, CMP immediate with memory should be encoded as "1000 00sw : mod 111 r/m : immediate data". Page B-12 POP "segment register CS, DS, ES" should be encoded as "segment register DS, ES".

P24. Machine-Check Initialization

The Intel Architecture Software Developer's Manual, Vol 3: System Programming Guide section 14.5 currently states:

14.5 Machine-Check Initialization

To use the processors machine-check architecture, software must initialize the processor to activate the machine-check exception and the error-reporting mechanism.

Example gives pseudocode for performing this initialization. This pseudocode checks for the existence of the machine-check architecture and exception on the processor, then enables the machine-check exception and the error-reporting register banks. The pseudocode assumes that the machine-check exception (#MC) handler has been installed on the system. This initialization procedure is compatible with the Pentium 4, Intel Xeon, P6 family, and Pentium processors.

Following power up or power cycling, the IA32_MCi_STATUS registers are not guaranteed to have valid data until after the registers are initially cleared to all 0s by software, as shown in the initialization pseudocode in the example.

Machine-Check Initialization Pseudocode

EXECUTE the CPUID instruction;

READ bits 7 (MCE) and 14 (MCA) of the EDX register;

IF CPU supports MCE

THEN

IF CPU supports MCA

THEN

IF IA32_ MCG_CAP.MCG_CTL_P = 1

(* IA32_MCG_CTL register is present *)

IA32_MCG_CTL ← FFFFFFFFFFFFFFFFH;

(* enables all MCA features *)

FI;

COUNT IA32_MCG_CAP.Count;

MAX_BANK_NUMBER ← COUNT˜ 1;

(* determine number of error-reporting banks supported *)

IF (P6 Family Processor)

THEN

FOR error-reporting banks (1 through MAX_BANK_NUMBER) DO

IA32_MCi_CTL ← FFFFFFFFFFFFFFFFH;

(* enables logging of all errors except for MC0_CTL register *)

OD

ELSE (* Pentium 4 and Intel Xeon Processors *)

FOR error-reporting banks (0 through MAX_BANK_NUMBER) DO

IA32_MCi_CTL ← FFFFFFFFFFFFFFFFH;

(* enables logging of all errors including MC0_CTL register *)

OD FI;

FOR error-reporting banks (0 through MAX_BANK_NUMBER) DO

IA32_MCi_STATUS ← 0000000000000000H; (* clears all errors *)

OD FI;

Set the MCE flag (bit 6) in CR4 register to enable machine-check exceptions;

FI;

It should state:

14.5 Machine-Check Initialization

To use the processors machine-check architecture, software must initialize the processor to activate the machine-check exception and the error-reporting mechanism.

Example gives pseudocode for performing this initialization. This pseudocode checks for the existence of the machine-check architecture and exception on the processor, then enables the machine-check exception and the error-reporting register banks. The pseudocode shown is compatible with the Pentium 4, Intel Xeon, P6 family, and Pentium processors.

Following power up or power cycling, the IA32_MCi_STATUS registers are not guaranteed to have valid data until after the registers are initially cleared to all 0s by software, as shown in the initialization pseudocode in Example . In addition, when using P6 family processors, the software must set MCi_STATUS registers to 0 when doing a soft-reset.

Machine-Check Initialization Pseudocode

```
Check CPUID Feature Flags for MCE and MCA support

IF CPU supports MCE

THEN

        IF CPU supports MCA

        THEN

                IF (IA32_MCG_CAP.MCG_CTL_P = 1)

                (* IA32_MCG_CTL register is present *)

                THEN

                        IA32_MCG_CTL ← FFFFFFFFFFFFFFFFH;

                        (* enables all MCA features *)

                FI

                (* Determine number of error-reporting banks supported *)


                COUNT IA32_MCG_CAP.Count;

                MAX_BANK_NUMBER ← COUNT - 1;


                IF (Processor Family is 6H)

                THEN

                        (* Enable logging of all errors except for MC0_CTL register *)

                        FOR error-reporting banks (1 through MAX_BANK_NUMBER)

                        DO

                                IA32_MCi_CTL ← 0FFFFFFFFFFFFFFFFH;

                        OD
```

```
        (* Clear all errors *)
        FOR error-reporting banks (0 through MAX_BANK_NUMBER)
        DO
                IA32_MCi_STATUS ← 0;
        OD

ELSE IF (Processor Family is 0FH) (*any Processor Extended Family *)
THEN
        (* Enable logging of all errors including MC0_CTL register *)
        FOR error-reporting banks (0 through MAX_BANK_NUMBER)
        DO
                IA32_MCi_CTL ← 0FFFFFFFFFFFFFFFFH;
        OD

        (* BIOS clears all errors only on power-on reset *)
        IF (BIOS detects Power-on reset)
        THEN
                FOR error-reporting banks (0 through MAX_BANK_NUMBER)
                DO
                        IA32_MCi_STATUS ← 0;
                OD
        ELSE
                FOR error-reporting banks (0 through MAX_BANK_NUMBER)
                DO
                        (Optional for BIOS and OS) Log valid errors
                        (OS only) IA32_MCi_STATUS ← 0;
                OD FI
        FI
FI
```

Setup the Machine Check Exception (#MC) handler for vector 18 in IDT

Set the MCE bit (bit 6) in CR4 register to enable Machine-Check Exceptions

FI

P25. Incorrect Description of MOVAPS #UD

The *Intel Architecture Software Developer's Manual, Vol 2: Instruction Set Reference* Section 3.2 "INSTRUCTION REFERENCE" MOVAPS #UD description is incorrect. Under Protected Mode Exceptions it currently states:

#UD If CPUID feature flag SSE2 is 0.

It should state:

#UD If CPUID feature flag SSE is 0.

Also, under Real-Address Mode Exceptions it currently states:

#UD If CPUID feature flag SSE2 is 0.

It should state:

#UD If CPUID feature flag SSE is 0.

P26. PSE-36 Paging Mechanism

The *Intel Architecture Software Developer's Manual, Vol 3: System Programming Guide* Chapter 5, Section 5.14, page 5-35 under “Interrupt 10 – Invalid TSS Exception (#TS)” the last paragraph of the description currently states:

To insure that a valid TSS is available to process the exception, the invalid TSS exception must be a task called using a task gate.

It should state:

The invalid-TSS handler must be a task called using a task gate. Handling this exception inside the faulting TSS context is not recommended and processor state may not be consistent

P27. Segment Wraparound compatibility

The *Intel Architecture Software Developer's Manual, Vol 3: System Programming Guide* Chapter 18, Section 18.29.1 (Segment Wraparound) has the following paragraph added at the end of the section:

The behavior when executing near the limit of a 4GB selector (limit=0xFFFFFFFF) is different between the Pentium Pro and the Pentium 4 family of processors. On the Pentium Pro, instructions which cross the limit -- for example, a two byte instruction such as INC EAX that is encoded as 0xFF 0xC0 starting exactly at the limit faults for a segment violation (a one byte instruction at 0xFFFFFFFF does not cause an exception). Using the Pentium 4 family, neither of these situations causes a fault

P28. Performance Counting Clocks

The *Intel Architecture Software Developer's Manual, Vol 3: System Programming Guide* Chapter 15, Section 15.9.9 (Counting Clocks) will be updated to clarify the impact of Intel Hyper-Threading Technology when considering clock ticks and the Time Stamp Counter. A number of updates have been integrated into the section.

P29. New Cascading Counters Chapter

The *Intel Architecture Software Developer's Manual, Vol 3: System Programming Guide* Chapter 15, Section 15.9.6.6 will be updated to include information on extended cascading, a new feature included in the Intel NetBurst microarchitecture.

P30. Updated Table B-1 in Appendix B,Vol.3

The *Intel Architecture Software Developer's Manual, Vol 3: System Programming Guide* Appendix B, a new column will be added to Table B-1“ MSRs in the Pentium 4 and Intel Xeon Processors”:

For Hyper-Thread enabled (HT) processors, there are two logical processors per physical unit. If an MSR is Shared, this means that one MSR is shared between two logical processors. If an MSR is Unique, this means that each logical processor has its own MSR. The new column identifies the characteristics of each MSR.

P31. Message Signaled Interrupts

In the *Intel Architecture Software Developer's Manual, Vol 3: System Programming Guide* Chapter 8, Section 8.11 will be edited to include information on Message Signaled Interrupts, an optional feature that enables PCI devices to request service by writing a system-specified message to a system-specified address (PCI DWORD memory write transaction).

P32. Selecting Memory Types for Pentium 4, Intel Xeon, and

Pentium III Processors

In the *Intel Architecture Software Developer's Manual, Vol 3: System Programming Guide* Chapter10, Section 10.5.2.2., Table 10.7, Effective Page-Level Memory Types for Pentium III, Pentium 4, and Intel Xeon Processors currently states:

Table 10-7. Effective Page-Level Memory Types for Pentium III, Pentium 4, and Intel Xeon Processors

MTRR Memory Type	PAT Entry Value	Effective Memory Type
UC	X	UC[1]
WC	UC	UC [2]
	UC-	WC
	WC	WC
	WT	Undefined
	WB	WC
	WP	Undefined
WT	UC	UC [2]
	UC-	UC [2]
	WC	WC
	WT	WT
	WB	WT
	WP	Undefined
WB	UC	UC [2]
	UC-	UC [2]
	WC	WC
	WT	WT
	WB	WB
	WP	WP
WP	UC	UC [2]
	UC	Undefined
	WC	WC
	WT	Undefined
	WB	WP
	WP	WP

NOTES:

1. The UC attribute comes from the MTRRs and the processors are not required to snoop their caches since the data could never have been cached. This attribute is preferred for performance reasons.

2. The UC attribute came from the page-table or page-directory entry and processors are required to check their caches because the data may be cached due to page aliasing, which is not recommended.

It should read

In the *Intel Architecture Software Developer's Manual, Vol 3: System Programming Guide* Chapter10, Section 10.5.2.2., Table 10.7, Effective Page-Level Memory Types for Pentium III, Pentium 4, and Intel Xeon Processors currently states:

Table 10-7. Effective Page-Level Memory Types for Pentium III, Pentium 4, and Intel Xeon Processors

MTRR Memory Type	PAT Entry Value	Effective Memory Type
UC	UC	UC[1]
WC	UC-	UC[1]
	WC	WC
	WT	UC[1]
	WB	UC[1]
	WP	UC[1]
	UC	UC[2]
WT	UC-	WC
	WC	WC
	WT	Undefined
	WB	WC
	WP	Undefined
	UC	UC [2]
WB	UC-	UC [2]
	WC	WC
	WT	WT
	WB	WT
	WP	Undefined
	UC	UC [2]
WP	UC-	UC [2]
	WC	WC
	WT	WT
	WB	WB
	WP	WP
	UC	UC [2]

NOTES:

1. The UC attribute comes from the MTRRs and the processors are not required to snoop their caches since the data could never have been cached. This attribute is preferred for performance reasons.

2. The UC attribute came from the page-table or page-directory entry and processors are required to check their caches because the data may be cached due to page aliasing, which is not recommended.

Specification Clarifications

P1. BR[3:2]# Signal Definition is not Reserved

The following apply to the *Intel® Xeon™ Processor at 1.40 GHz, 1.50 GHz. 1.70 GHz and 2 GHz Datasheet* and *Intel® Xeon™ Processor with 512 KB L2 Cache at 1.80 GHz to 2.40 GHz Datasheet*.

In dual-processor (DP) systems that use DP processors, the BR[3:2]# signals are not truly "reserved". The processor will hang if these signals are left floating or if these processors are installed in a 4P system. These signals require baseboard pull-ups to processor Vcc on DP-only platforms. Datasheets include a note stating this but it should be clarified by removing the "reserved" comment as follows:

Notes:

1. In systems utilizing the Intel® Xeon™ processor, the system designer must pull-up these signals to the processor V_{CC}.

.

This page is intentionally left blank.

Documentation Changes

There are no documentation changes in this Specification Update revision.